PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-112901
(43)Date of publication of application : 21.04.2000

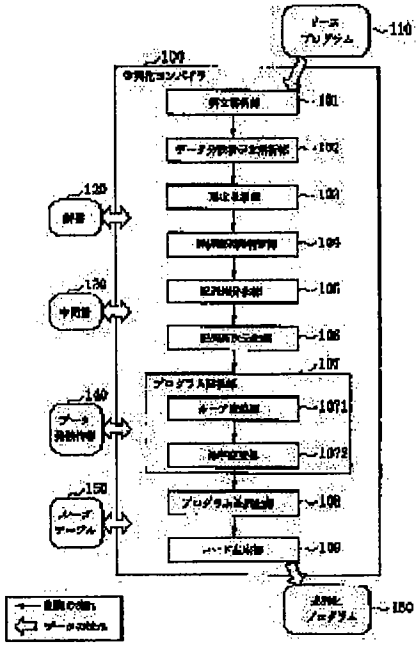
(51)Int.Cl. G06F 15/16
G06F 9/45

(21)Application number : 10-281692
(22)Date of filing : 02.10.1998

(71)Applicant : HITACHI LTD
REAL WORLD COMPUTING PARTNERSHIP
(72)Inventor : SATO MAKOTO
HIROOKA TAKASHI

(54) DATA REARRANGING METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To eliminate the need of a copy in data re-distribution while a program is executed and to speed up the execution of the program in a distributed storage-type or distributed common storage-type parallel machine.
SOLUTION: A data distribution form showing the allocation method of respective array elements to a distributed memory is collected (form collection part 103). Respective arrays are divided into sets formed of partial arrays belonging to a unique processor in any data distribution form (array thinning part 105). The number of the dimensions of the original array is increased so that the array elements in the respective partial arrays become continuous addresses and the order of the array elements is rearranged (array high-dimension forming part 106). It is judged that the number of the types of the data distribution forms in the respective dimensions of array is not more than two (form type number judgement 104). It is detected that a loop control variable is contained in the added character of the dimension which is made to be higher in array and a loop having the loop control variable is converted into plural loops nested in accordance with the forming of the array dimension to be high (loop conversion par 1071). The array added character in an array reference point in a program is changed in accordance with the forming of the array dimension of array added character at an array reference point in a program to be high (added character change part 1072).



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－112901
（P2000－112901A）
(43)公開日　平成12年４月21日(2000.4.21)

(51)Int.Cl.[7]	識別記号	ＦＩ		テーマコード*（参考）
Ｇ０６Ｆ　15/16	６２０	Ｇ０６Ｆ　15/16	６２０Ｈ	５Ｂ０４５
9/45		9/44	３２２Ｆ	５Ｂ０８１

審査請求　未請求　請求項の数８　ＯＬ　（全 34 頁）

(21)出願番号　特願平10－281692

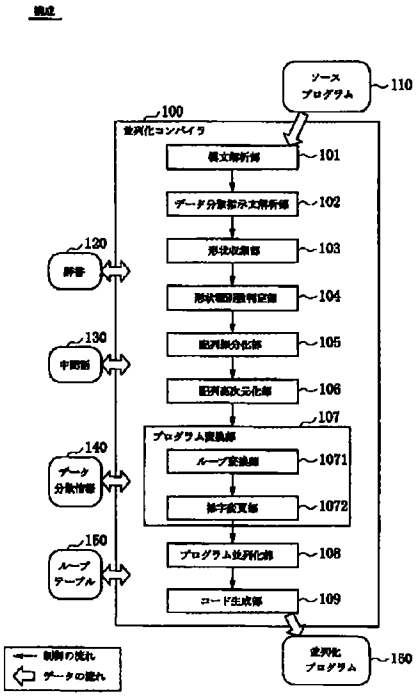
(22)出願日　平成10年10月２日(1998.10.2)

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(71)出願人　593162453
技術研究組合新情報処理開発機構
東京都千代田区東神田２－５－12　龍角散ビル８階
(72)発明者　佐藤　真琴
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内
(74)復代理人　100102587
弁理士　渡邉　昌幸　（外１名）

最終頁に続く

(54)【発明の名称】　データ再配置方法

(57)【要約】
【課題】　分散記憶型または分散共有記憶型並列機で、プログラム実行中のデータ再分散におけるコピーを不要にしプログラムの実行を高速化すること。
【解決手段】　各配列要素の分散メモリへの割付け方法を表わすデータ分散形状を収集し(形状収集部103)、各配列をどのデータ分散形状においても各部分配列が唯一のプロセッサに属するような部分配列からなる集合に分割し(配列細分化部105)、各部分配列内の配列要素が連続アドレスになるように元の配列の次元数を増加させて配列要素の順序を並び替え(配列高次元化部106)、配列の各次元のデータ分散形状の種類が２種類以下であることを判定し(形状種別数判定104)、配列の高次元化される次元の添字にループ制御変数が含まれることを検出し、そのループ制御変数を持つループを配列高次元化に対応してネストする複数のループに変換し(ループ変換部1071)、プログラム中の配列参照点における配列添字を配列高次元化に対応して変更する(添字変更部1072)。



1

【特許請求の範囲】

【請求項１】　逐次計算機向けのソースプログラムまたはデータ分散指示文あるいはデータ再分散指示文を含む並列計算機向けのソースプログラムを入力し、並列計算機の各分散メモリにデータを分散し、データ再分散指示文またはデータ再分散ルーチンを含む並列計算機向けソースプログラムまたはオブジェクトプログラムを出力するコンパイラにおけるデータ再配置方法であって、

プログラム中の各点における、各配列要素の分散メモリへの割付け方法を表わすデータ分散形状を収集する形状収集ステップと、

各配列を、上記形状収集ステップで得られたどのデータ分散形状においても、各部分配列が唯一のプロセッサに属するような部分配列からなる集合に分割する配列細分化ステップと、

上記各部分配列内の配列要素が連続アドレスになるように元の配列の次元数を増加させて配列要素の順序を並び替える配列高次元化ステップを有することを特徴とするデータ再配置方法。

【請求項２】　逐次計算機向けまたは並列指示文を含む並列計算機向けのソースプログラムを入力し、共有メモリ型並列計算機向けの並列化指示文を含むソースプログラムまたはオブジェクトプログラムを出力するコンパイラにおけるデータ再配置方法であって、

複数プロセッサがその部分配列を分担して計算すべき配列に対して、プログラム中の各点における、各配列要素のプロセッサへの割付け方法を表わすデータ分担形状を収集する形状収集ステップと、

各配列を、上記形状収集ステップで得られたどのデータ分担形状においても各部分配列が唯一のプロセッサの分担になるような部分配列からなる集合に分割する配列細分化ステップと、

上記各部分配列内の配列要素が連続アドレスになるように元の配列の次元数を増加させて配列要素の順序を並び替える配列高次元化ステップを有することを特徴とするデータ再配置方法。

【請求項３】　請求項１または２記載のデータ再配置方法において、上記形状収集ステップによって得られた配列の各次元のデータ分散形状またはデータ分担形状の種類が２種類以下であることを判定する形状種別数判定ステップをさらに有することを特徴とするデータ再配置方法。

【請求項４】　請求項１または２記載のデータ再配置方法において、上記配列高次元化ステップは、元の配列の各要素を、それぞれの請求項における配列細分化ステップによって得られた部分配列中の要素と各部分配列の全体配列中での位置を示す位置配列の要素との組で表現可能なように、元の配列から、上記部分配列の次元数と上記位置配列の次元数の和の次元数を持つ高次元化配列を作成し、上記高次元化配列の連続アドレスを与える次元から上記部分配列の次元数分の次元までの各寸法は、上記部分配列の各次元の寸法とし、残りの次元の寸法は、上記位置配列の各次元の寸法とすることを特徴とするデータ再配置方法。

2

【請求項５】　請求項１または２記載のデータ再配置方法において、上記配列高次元化ステップに伴い、配列の高次元化される次元の添字にループ制御変数が含まれることを検出し、そのループ制御変数を持つループを、配列高次元化ステップに対応してネストする複数のループに変換するループ変換ステップと、プログラム中の配列参照点における配列添字を、配列高次元化ステップに対応して変更する添字変更ステップとをさらに含むことを特徴とするデータ再配置方法。

【請求項６】　請求項５記載のデータ再配置方法において、上記ループ変換ステップは、配列要素の参照が部分配列間の境界をまたがるか否かを判定し、境界をまたがる場合とまたがらない場合とで別ループになるようにループの繰り返し範囲を分割するための解析を行なうループピーリング解析ステップと、上記解析の結果を用いてループの繰返し範囲を分割するループピーリングステップとを含むことを特徴とするデータ再配置方法。

【請求項７】　請求項１記載のデータ再配置方法において、通信データのバッファリングを行なわない再分散通信を生成する非バッファリング再分散通信生成ステップをさらに含むことを特徴とするデータ再配置方法。

【請求項８】　請求項７記載のデータ再配置方法において、分散メモリ型並列計算機の各プロセッサで、データ分散前の元の配列全体を宣言するグローバルデータ宣言作成ステップと、各配列参照点における配列添字の値を、データ分散前の元の値を用いるグローバル添字生成ステップをさらに含むことを特徴とするデータ再配置方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、逐次計算機向けのプログラムまたは並列計算機向けのデータ再分散指示を含むプログラムを入力してデータ再配置を行なったり、あるいは、データ再配置指示を挿入した並列計算機向けのプログラムまたはオブジェクトプログラムを生成するコンパイラに係り、特に、バッファへのコピー時間を不要とし、プログラムの実行時間を短縮することが可能なデータ再配置方法に関する。

【０００２】

【従来技術】従来、分散メモリ型並列計算機におけるデータ再分散を高速化する技術として、ダニエル・ジェイ・パレルモ、オイゲネ・ヴェー・ホッジ・フィエル、プリスヴィラージ・バナジー著「ダイナミック データ パーティショニング フォー ディストリビューテッド－メモリ マルチコンピューターズ」ジャーナル オブ パラレル アンド ディストリビューテッド コンピューティ

ング ３８、第１５８頁から第１７５頁（１９８８）（Daniel J. Palermo, Eugene W. Hodges IV, andPrithviraj Banerjee. "Dynamic Data Partitioning for Distributed-Memory Multicomputers", Journal of Parallel and Distributed Computing 38, pp. 158-175 (1998)）中のpp.170-171で言及されているように、データ再分散オペレーションのアグレゲーション、即ち、データ再分散すべき複数の配列、または、データ再分散すべき一つの配列の中の非連続な部分を、１つのバッファの連続アドレス領域へプログラム実行時にコピーし、そのバッファに対して再分散を唯１回、行なうことで、再分散の起動オーバヘッドを削減するものがあった。

【０００３】また、従来、共有メモリ型並列計算機におけるFalse Sharingと呼ばれるキャッシュ間コンフリクトを削減する技術として、ジェニファー・エム・アンダースン、サマン・ピー・アマラシンハ、モニカ・エス・ラム著「データ アンド コミュニケーション トランスフォーメーションズ フォー マルチプロセッサーズ」ピーピーオーピーピー９５、第１６６頁カラ第１７８頁、１９９５（Jennifer M. Anderson, Saman P. Amarasinghe, and Monica S. Lam. "Data and Communication Transformations for Multiprocessors", PPOPP'95, pp. 166-178 (1995)）で述べられているように、配列の次元の内、アドレスが連続する配列要素を特定するのに使われる次元を、プロセッサ個数分に分割し、その次元を、一つのプロセッサがアクセスする添字だけからなる次元と、プロセッサ番号を指定する次元の２つの次元に分割し、後者を、アドレスの増加が最も遅い次元になるように配列を高次元化することで、キャッシュ内のデータが２つのプロセッサにまたがる機会を減少させるものがあった。

【０００４】

【発明が解決しようとする課題】上記第１の従来技術は、プログラム実行時にバッファへコピーするため、コピーに余分な時間がかかるという問題点がある。また、上記第１の従来技術は、再分散前後で配列の大きさが変化する可能性があるため、再分散のたびに配列領域の確保・開放が必要で、それらに余分な時間がかかるという問題点がある。さらに、上記第１の従来技術は、上記配列領域の確保・開放、および、再分散前後で配列の形状が変化することにより、本来は不要な、再分散前後で同一のプロセッサに割り当てられる部分配列のコピーも必要で、それに余分な時間がかかるという問題点がある。

【０００５】上記第２の従来技術は、ある特定の次元の配列要素を複数プロセッサが分担して計算する時には適用可能だが、配列が３次元以上で、プログラム中に２つの異なるループが存在し、複数プロセッサが分担して計算する配列の次元がそれらのループに対して、各々、アドレスが連続になる次元から１次元目と２次元目である時、前者のループに対してはキャッシュ間コンフリクトの一種であるフォールスシェアリングがかなり削減可能である。しかし、後者のループに対しては配列の２次元目の、プロセッサ間の割り当て境界で、３次元目の各添字に対してプロセッサ総数だけ両プロセッサがキャッシュ共有する可能性があるため、フォールスシェアリングはあまり削減できないという問題点がある。

【０００６】本発明の目的は、プログラム実行時にバッファへのコピー処理をなくし、プログラムの実行時間を短縮することである。また、本発明の別の目的は、再分散時の、配列領域の確保・開放処理をなくし、プログラムの実行時間を短縮することである。また、本発明の別の目的は、再分散前後で同一のプロセッサに割り当てられる部分配列のコピーをなくすことにより、プログラムの実行時間を短縮することである。また、本発明の別の目的は、プログラム中に複数の異なるループが存在し、各々のループで、複数プロセッサが分担して計算する配列の次元が異なる時に、それら全てのループに対してキャッシュ間コンフリクトを削減することにより、プログラムの実行時間を短縮することである。

【０００７】

【課題を解決するための手段】本発明は、上記目的を達成するために、プログラム中の各点における、各配列要素の分散メモリへの割付け方法を表わすデータ分散形状を収集する形状収集ステップ（図１の形状収集部１０３）と、各配列を、上記形状収集で得られたどのデータ分散形状においても、各部分配列が唯一のプロセッサに属するような部分配列からなる集合に分割する配列細分化ステップ（同、配列細分化部１０５）と、上記各部分配列内の配列要素が連続アドレスになるように元の配列の次元数を増加させて配列要素の順序を並び替える配列高次元化ステップ（同、配列高次元化部１０６）と、配列の各次元のデータ分散形状の種類が２種類以下であることを判定する形状種別数判定ステップ（同、形状種別数判定１０４）と、配列の高次元化される次元の添字にループ制御変数が含まれることを検出し、そのループ制御変数を持つループを、配列高次元化に対応して、ネストする複数のループに変換するループ変換ステップ（同、ループ変換部１０７１）と、プログラム中の配列参照点における配列添字を、配列高次元化に対応して変更する添字変更ステップ（同、添字変更部１０７２）と、分散メモリ型並列計算機の各プロセッサで、データ分散前の元の配列全体を宣言するグローバルデータ宣言作成ステップ（図７のステップ７０６）と、各配列参照点における配列添字の値を、データ分散前の元の値を用いるグローバル添字生成ステップ（図７のステップ７０５２）と、通信データのバッファリングを行なわない再分散通信を生成する非バッファリング再分散通信生成ステップ（図７のステップ７０４１）を含むものである。

【０００８】また、本発明は、上記別の目的を達成するために、複数プロセッサがその部分配列を分担して計算

すべき配列に対して、プログラム中の各点における、各配列要素のプロセッサへの割付け方法を表わすデータ分担形状を収集する形状収集ステップ（図２７の形状収集部１０３）と、各配列を、上記形状収集で得られたどのデータ分担形状においても、各部分配列が唯一のプロセッサの分担になるような部分配列からなる集合に分割する配列細分化ステップ（同、配列細分化部１０５）と、上記各部分配列内の配列要素が連続アドレスになるように元の配列の次元数を増加させて配列要素の順序を並び替える配列高次元化ステップ（同、配列高次元化部１０６）と、配列の各次元のデータ分担形状の種類が２種類以下であることを判定する形状種別数判定ステップ（同、形状種別数判定部１０４）と、配列の高次元化される次元の添字にループ制御変数が含まれることを検出し、そのループ制御変数を持つループを、配列高次元化に対応して、ネストする複数のループに変換するループ変換ステップ（同、ループ変換部１０７１）と、プログラム中の配列参照点における配列添字を、配列高次元化に対応して変更する添字変更ステップ（同、添字変更部１０７２）と、分散メモリ型並列計算機の各プロセッサで、データ分散前の元の配列全体を宣言するグローバルデータ宣言作成ステップ（図７のステップ７０６）と、各配列参照点における配列添字の値を、データ分散前の元の値を用いるグローバル添字生成ステップ（図７のステップ７０５２）と、通信データのバッファリングを行なわない再分散通信を生成する非バッファリング再分散通信生成ステップ（図７のステップ７０４１）を含むものである。

【０００９】

【発明の実施の形態】（第１の実施例）以下、本発明の第１の実施例を図１から図２６を用いて説明する。第１の実施例では分散メモリを持つ並列計算機を対象としている。図１は、本発明による並列化コンパイラの構成を示したものである。１００は並列化コンパイラである。並列化コンパイラ１００は、同図に示すように、ソースプログラム１１０を入力して辞書１２０及び中間語１３０を出力する構文解析部１０１、辞書１２０及び中間語１３０を入力してデータ分散指示文を解析し、データ分散情報１４０を出力するデータ分散指示文解析部１０２、データ分散情報１４０を入力してプログラム内に現れるデータ分散形状情報を収集し、その結果を辞書１２０に反映させる形状収集部１０３、辞書１２０及びデータ分散情報１４０を入力して配列のデータ分散の種別数を判定し、配列を細分化すべきか否かを辞書１２０に反映させる形状種別数判定部１０４、辞書１２０及びデータ分散情報１４０を入力して配列を細分化し、その結果を辞書１２０に反映させる配列細分化部１０５、辞書１２０及びデータ分散情報１４０を入力して、データ分散形状が変化する次元の配列添字範囲を分割して、その次元を２つの次元に分けることによって配列を高次元化し、その結果を辞書１２０に反映させる配列高次元化部１０６、辞書１２０，データ分散情報１４０及び中間語１３０を入力してプログラムを変換し、その結果を中間語１３０に反映し、またループテーブル１５０を出力するプログラム変換部１０７，辞書１２０，データ分散情報１４０，中間語１３０及びループテーブル１５０を入力してプログラムを並列化し、その結果を辞書１２０，中間語１３０，ループテーブル１５０に反映するプログラム並列化部１０８、辞書１２０，中間語１３０，ループテーブル１５０を入力して並列化プログラム１６０を出力するコード生成部１０９よりなる。

【００１０】ここで、プログラム変換部１０７は、辞書１２０，データ分散情報１４０及び中間語１３０を入力して、配列を高次元化する時に添字範囲を分割することによって生じる、分割境界をまたいだ配列要素の参照とそうでない配列要素の参照で添字が異なることに対処するために、分割境界をまたいだ配列要素を参照する文を含むループ繰り返しと、そのような文を含まないループ繰り返しを別のループにするループピーリングと、配列高次元化に伴い、増加した添字に対するループを作成するループ多重化とを行い、その結果をループテーブル１５０を作成してそこに反映し、また中間語１３０に反映するループ変換部１０７１、配列の高次元化とループピーリングに伴って、配列の添字を変更する添字変更部１０７２、よりなる。

【００１１】本実施例におけるソースプログラム１１０は、ＨＰＦ(High Performance Fortran)のようなデータ分散指示文の入ったプログラムであり、並列化プログラム１６０は、分散メモリマシン向けのＭＰＩ(Message Passing Interface)のようなメッセージ通信入りの並列化プログラムである。ここで、データ分散指示文とは、データを分割して分割された個々のデータを複数の論理プロセッサの内の一つに割り付けるように、コンパイラに指示する文である。

【００１２】ＨＰＦの言語仕様の説明は、ハイ パフォーマンス フォートラン フォーラム編「ハイ パフォーマンス フォートラン ランゲージ スペシフィケーション バージョン2.0.α.2」センター フォー リサーチ オン パラレル コンピュテーション、ライス ユニバーシティ、ヒューストン、テキサス、１９９６ (High Performance Fortran Forum, "High Performance Fortran Language SpecificationVer. 2.0.α.2", Center for Research on Parallel Computation, Rice Univ.,Houston, Tx, 1996.) に詳しい。

【００１３】ＭＰＩの仕様の説明は、メッセージ パッシング インターフェース フォーラム編「エム ピー アイ：ア メッセージ ―パッシング インターフェース スタンダード」ユニバーシティ オブ テネシー、ノックスビル、テネシー、１９９４ (Message Passing Interface Forum, "MPI: A Message-Passing Interface Standar

d", University of Tennessee, Knoxville, Tennessse, 1994.）に詳しい。

【００１４】次に、本発明の並列化コンパイラの動作を、図１を元に、図２から図２６まで、具体例を用いて説明する。図８はソースプログラム１１０の具体例である。文８００は、Ｆｏｒｔｒａｎの文法による配列ａ及びスカラ変数ｂ，ｃの宣言である。文８０３と文８１７で囲まれた部分は、ｋをループ制御変数とするループ本体である。文８０５と文８０９で囲まれた部分は、ｊをループ制御変数とするループ本体であり、文８０６と文８０８で囲まれた部分は、ｉをループ制御変数とするループ本体である。文８１１と文８１５で囲まれた部分も同様に、ｊをループ制御変数とするループ本体であり、文８１２と文８１４で囲まれた部分も同様に、ｉをループ制御変数とするループ本体である。

【００１５】文８１６は、スカラ変数ｂ，ｃの値を変更する手続き呼出し文である。文８０１，文８０２，文８０４，及び文８１０はＨＰＦの指示文である。“！ＨＰＦ＄”は、それで始まる文がＨＰＦの指示文であることを表わすキーワードである。文８０１は論理プロセッサの配置形状を指定する指示文である。“ＰＲＯＣＥＳＳＯＲＳ”はキーワードであり、それに続く“ｐｒｏｃ（４）”により、論理プロセッサ４台が直線状に配置されていることを宣言している。文８０２は、配列ａの初期分割を指示する文である。“ＤＩＳＴＲＩＢＵＴＥ”はキーワードであり、それに続く“ａ（ｂｌｏｃｋ，＊）　ＯＮＴＯ　ｐｒｏｃ”により、配列ａの１次元目をｂｌｏｃｋという方法で分割し、分割された各々の部分配列を論理プロセッサｐｒｏｃ（１），ｐｒｏｃ（２），ｐｒｏｃ（３），ｐｒｏｃ（４）に、それぞれ割り付けることを表わす。

【００１６】ここで、上記の論理プロセッサ宣言において、配列ａの１次元目をｂｌｏｃｋ分割するとは、配列ａの１次元目の下限値を１、上限値をＮとすると、配列ａの１次元目の添字を、１から［（Ｎ－１）／４］＋１個ずつ区切って、ｐｒｏｃ（１）から順番に割り付けることである。今の場合、ａ（１：３，１：１２），ａ（４：６，１：１２），ａ（７：９，１：１２），ａ（１０：１２，１：１２）がそれぞれ、ｐｒｏｃ（１），ｐｒｏｃ（２），ｐｒｏｃ（３），ｐｒｏｃ（４）に割り付けられる。

【００１７】文８０４は配列ａをプログラム実行中に再分割することを指示する文である。“ＲＥＤＩＳＴＲＩＢＵＴＥ”はキーワードであり、それに続く“ａ（ｂｌｏｃｋ，＊）　ＯＮＴＯ　ｐｒｏｃ”により、配列ａの１次元目をｂｌｏｃｋという方法で分割し、分割された各々の部分配列を論理プロセッサｐｒｏｃ（１），ｐｒｏｃ（２），ｐｒｏｃ（３），ｐｒｏｃ（４）に、それぞれ割り付けることを表わす。この指示文により、それまで他の方法で論理プロセッサに割り付けられていた配列要素は、配列ａの１次元目をｂｌｏｃｋ分割するような割り付けられ方に変更される。これにより、一般に論理プロセッサ間のデータ移動が発生する。

【００１８】文８１０も同様に配列ａをプログラム実行中に再分割することを指示する文である。この文は、文８０４とは異なり、配列ａの２次元目をｂｌｏｃｋ分割するような割り付けかたを指示する。今の場合、ａ（１：１２，１：３），ａ（１：１２，４：６），ａ（１：１２，７：９），ａ（１：１２，１０：１２）がそれぞれ、ｐｒｏｃ（１），ｐｒｏｃ（２），ｐｒｏｃ（３），ｐｒｏｃ（４）に割り付けられる。

【００１９】以下、並列化コンパイラ１００の中の個々の処理を説明する。構文解析部１０１は上記ソースプログラム１１０を入力して辞書１２０、中間語１３０を生成する。中間語１３０はソースプログラム１１０に対応しているので、以下の説明では、図８のソースプログラム１１０を、中間語１３０のソースプログラムイメージの表現として用いる。

【００２０】図１０は、構文解析部１０１が生成する辞書１２０を示したものである。１０００は、配列ａに対する辞書１２０である。本説明ではスカラ変数ｂとｃに対する辞書は説明で用いないため、省略する。１００１は変数名を示し、配列名である“ａ”という文字列が設定されている。１００２は変数の型で、実数型を示す“ＲＥＡＬ”が設定されている。１００３は変数長で、４バイト変数を示す“４”が設定されている。１００４は変数が配列か否かを表わすフラグで、ａは配列なので“ＴＲＵＥ”が設定されている。

【００２１】１００５は配列を細分化すべきか否かを表わすフラグであり、“ＴＲＵＥ”は配列を細分化すべきであることを表わす。この値は形状種別数判定１０４で設定される。１００６は変数が配列の場合、その次元数を表わす。配列ａは２次元配列なので、値２が設定されている。１００７は変数が配列の場合、その配列の各次元の寸法宣言情報を格納した配列辞書１０４０を指すポインタである。１００８は細分化情報１１００へのポインタである。これは配列細分化１０５で設定される。１００９は高次元化辞書１２００へのポインタである。これは配列高次元化１０６で設定される。１０１０はデータ分散情報の収集リストの先頭１０２０へのポインタである。これは形状収集１０３で設定される。

【００２２】１０４０は配列の各次元の寸法宣言情報を格納した配列辞書であり、辞書１０００のフィールド１００７から指されている。１０４１は配列ａの第１次元目の下限値と上限値が、各々、１と１２であることを表わしている。１０４２は配列ａの第２次元目の下限値と上限値が、各々、１と１２であることを表わしている。

【００２３】データ分散指示文解析１０２は、辞書１２０及び中間語１３０を入力して、データ分散情報１４０を出力する。図９は、データ分散情報１４０を示したも

のである。上記説明した通り、中間語１３０として図８のソースプログラム１１０を用いる。また、図８における文８０２に対するデータ分散情報は文８０４に対するデータ分散情報と同じなので、ここでは文８０４と文８１０に対するデータ分散情報、及び文８０１に対する論理プロセッサ情報のみ示す。

【００２４】９４０はＰＲＯＣＥＳＳＯＲＳ指示文８０１を解析して得たＰＲＯＣＥＳＳＯＲＳ情報テーブルである。９４１は論理プロセッサの変数名を表わす文字列“ｐｒｏｃ”である。９４２は論理プロセッサの配列次元数であり、値１は１次元配列であることを示す。９４３は論理プロセッサの各次元の寸法情報を表わす配列辞書９５０へのポインタである。９５０は論理プロセッサの各次元の寸法情報を表わす配列辞書であり、論理プロセッサｐｒｏｃの第１次元目の下限値と上限値が、各々、０と３であることを示している。

【００２５】９００は再分散指示文８０４を解析して得た、配列ａに対するデータ分散情報テーブルである。９０１は８０４の指示文に対してデータ分散情報が変化する他の変数へのポインタである。今の場合、そのような変数は存在しないので値ＮＵＬＬが設定されている。９０２は配列ａの次元数であり、値２が設定されている。９０３は各次元ごとのデータ分散情報を表わす次元毎データ分散情報テーブル９２０へのポインタである。９０４は論理プロセッサに対するＰＲＯＣＥＳＳＯＲＳ情報テーブルへのポインタであり、ｐｒｏｃに対するＰＲＯＣＥＳＳＯＲＳ情報テーブル９４０を指している。

【００２６】９２０は配列ａに対する次元毎データ分散情報テーブルである。９２１は第１次元目の分散形状がｂｌｏｃｋ分割であることを示している。９２２は第１次元目のｂｌｏｃｋ分割により、連続する３個の配列要素が各論理プロセッサに割り当てられることを示している。この数値３は、配列辞書１０４０から得られる配列ａの第１次元目の寸法「１２」を、配列辞書９５０から得られる論理プロセッサ数「４」で割って、小数点以下の端数を切り上げることで得られる。今の場合は「３」となる。この計算方法は前出の“High PerformanceFortran Language Specification”に記述されている。９２３は第２次元目が分散されないことを示している。９２４は第２次元目の、連続する１２個の配列要素が各論理プロセッサに割り当てられることを示している。即ち、これは第２次元目の全部が各論理プロセッサに割り当てられることを示している。

【００２７】９１０は再分散指示文８１０を解析して得た、配列ａに対するデータ分散情報テーブルである。９１１は再分散指示文８１０に対してデータ分散情報が変化する他の変数へのポインタである。今の場合、そのような変数は存在しないので値NULLが設定されている。９１２は配列ａの次元数であり、値２が設定されている。９１３は各次元ごとのデータ分散情報を表わす次元毎データ分散情報テーブル９３０へのポインタである。

【００２８】９１４は論理プロセッサに対するＰＲＯＣＥＳＳＯＲＳ情報テーブルへのポインタであり、ｐｒｏｃに対するＰＲＯＣＥＳＳＯＲＳ情報テーブル９４０を指している。９３０は配列ａに対する次元毎データ分散情報テーブルである。９３１は第１次元目が分散されないことを示している。９３２は第１次元目の、連続する１２個の配列要素が各論理プロセッサに割り当てられることを示している。即ち、これは第２次元目の全部が各論理プロセッサに割り当てられることを示している。９３３は第２次元目の分散形状がｂｌｏｃｋ分割であることを示している。９３４は第２次元目のｂｌｏｃｋ分割により、連続する３個の配列要素が各論理プロセッサに割り当てられることを示している。この数値３の計算方法は９２２の場合と同様である。

【００２９】データ分散指示文解析部１０２は、上記説明したデータ分散情報テーブル９００及び９１０を得るものであり、通常の構文解析処理に、辞書の各次元ごとの寸法と論理プロセッサ数から９２２及び９３４を計算する処理を加えただけのものであるので、詳細なアルゴリズムは省略する。

【００３０】次に、形状収集部１０３はデータ分散情報１４０を入力して、変数ごとのデータ分散情報をまとめるための収集リストを作成して、その収集リストの先頭を変数の辞書１２０に接続する。図１０の１０２０と１０３０はその結果得られるデータ分散情報の収集リストであり、１０２０はその内の先頭であり、辞書１０００のフィールド１０１０から指されている。１０２１は次の収集リストへのポインタであり、次の収集リスト１０３０を指している。１０２２は形状収集１０３が収集するデータ分散情報９００へのポインタである。１０３１は次の収集リストへのポインタであり、もうリストがないので、値ＮＵＬＬが設定されている。１０３２は形状収集１０３が収集するデータ分散情報９１０へのポインタである。

【００３１】形状種別数判定部１０４は、辞書１２０及びデータ分散情報１４０を入力して配列のデータ分散の種別数を判定し、種別数が２の時は配列を細分化すべきと判定し、その結果を辞書１２０に反映させる。まず、配列ａに対する辞書１０００のフィールド１０１０から収集リスト１０２０及び１０３０をたどり、そこから得られるデータ分散情報テーブル９００及び９１０が各々異なる分散形状（ｂｌｏｃｋ，＊），（＊，ｂｌｏｃｋ）であることからデータ分散の種別数２を得る。その結果、細分化すべきと判定し、細分化フラグ１００５の値をＴＲＵＥに設定する。

【００３２】配列細分化部１０５は、辞書１２０及びデータ分散情報１４０を入力して配列を細分化し、その結果を辞書１２０に反映させる。図２は配列細分化部１０５の処理手順を示したものである。以下、配列ａに対す

る処理を示す。まず、形状種別判定部１０４で設定された１００５のフラグの値がＴＲＵＥなので、配列ａは細分化すべきであり、ステップ２００はＹｅｓとなる。次に、ステップ２０１でＩの値が１となる。収集リスト１０２０と１０３０から、配列ａのデータ分散情報テーブル９００、９１０をたどり、それらから指される９２１及び９３１より、配列ａの１次元目のデータ分散形状が“ｂｌｏｃｋ”と“＊”とわかる。今の場合、２０２におけるブロック分割の記法におけるｍの値は、９２２の値３にあたるので、ステップ２０２はＹｅｓとなる。

【００３３】ステップ２０３では、細分化フラグＴＲＵＥ、細分分散ｂｌｏｃｋ、細分幅ｍとして３、細分ＰＥ数として１次元目がｂｌｏｃｋ分散となるデータ分散指示文８０４の、ｂｌｏｃｋに対する論理プロセッサ数４を、細分化情報の１次元目に設定する。

【００３４】図１１は細分化情報１１００を示したものである。１１０１は１次元目に対する細分化フラグの値を、１１０２は１次元目に対する細分分散を、１１０３は１次元目に対する細分幅を、１１０４は１次元目に対する細分ＰＥ数を示している。同様にして、１１０５は２次元目に対する細分化フラグの値を、１１０６は２次元目に対する細分分散を、１１０７は２次元目に対する細分幅を、１１０８は２次元目に対する細分ＰＥ数を示す。今、Ｉが１なので、ステップ２０３で設定された情報は細分化情報１１００の１次元目に対するフィールドに格納される。即ち、細分化フラグの値ＴＲＵＥが１１０１に、細分分散ｂｌｏｃｋが１１０２に、細分幅３が１１０３に、細分ＰＥ数４が１１０４に格納される。

【００３５】次に、ステップ２０７において、Ｉの値が２になる。配列ａは２次元なのでステップ２０８はＮｏとなり、処理はステップ２０２に戻る。収集リスト１０２０と１０３０から、配列ａのデータ分散情報テーブル９００及び９１０をたどり、それらから指される９２３，９３３より、配列ａの２次元目のデータ分散形状が“＊”と“ｂｌｏｃｋ”とわかる。今の場合、ステップ２０２におけるブロック分割の記法におけるｍの値は、９３４の値３にあたるので、ステップ２０２はＹｅｓとなる。

【００３６】ステップ２０３では、細分化フラグＴＲＵＥ、細分分散ｂｌｏｃｋ、細分幅ｍとして３、細分ＰＥ数として２次元目がｂｌｏｃｋ分散となるデータ分散指示文８１０の、ｂｌｏｃｋに対する論理プロセッサ数４を、細分化情報の２次元目に設定する。今、Ｉが２なので、ステップ２０３で設定された情報は細分化情報１１００の２次元目に対するフィールドに格納される。即ち、細分化フラグの値ＴＲＵＥが１１０５に、細分分散ｂｌｏｃｋが１１０６に、細分幅３が１１０７に、細分ＰＥ数４が１１０８に格納される。

【００３７】次に、ステップ２０７において、Ｉの値が３になる。配列ａは２次元なのでステップ２０８はＹｅｓになる。他にデータ分散配列はないので、ステップ２０９はＮｏとなり処理を終了する。

【００３８】次に、配列高次元化部１０６は、辞書１２０及びデータ分散情報１４０を入力して配列宣言を高次元化することにより配列要素のアドレス順序を変更し、その結果を辞書１２０に反映させる。図３は配列高次元化部１０６の処理手順を示したものである。以下、配列ａに対する処理を示す。

【００３９】まず、形状種別判定部１０４で設定された１００５のフラグの値がＴＲＵＥなので、配列ａは細分化すべきであり、ステップ３００はＹｅｓとなる。ステップ３０１では、配列ａの辞書１０００のフィールド１００６より配列の次元数Ｋは２、細分化情報１１００のフラグ１１０１と１１０５が両方ともＴＲＵＥなので、細分化すべき次元数Ｊは２で、Ｋ＋Ｊ＝４となって、４次元分の高次元化辞書の領域を確保する。

【００４０】図１２は高次元化辞書１２００を示したものである。１２０１は高次元化後の配列の次元数、１２０２は高次元化後の配列の次元毎情報テーブル１２１０へのポインタ、１２１０は次元毎情報テーブル、１２１１，１２１４，１２１７，１２２０は各々、高次元化後の配列の１，２，３，４次元目の高次元化フラグである。高次元化フラグには、高次元化の対象次元なら“ｃｈａｎｇｅｄ”が、高次元化により新しく作成される次元なら“ｎｅｗ”が、高次元化の対象でない次元なら“ｎｏｔ ｃｈａｎｇｅｄ”が設定される。

【００４１】１２１２，１２１５，１２１８，１２２１は各々、高次元化後の配列の１，２，３，４次元目の高次元化対応次元である。高次元化対応次元には、高次元化の対象次元なら、その次元に対応する新しく作成される次元が、高次元化により新しく作成される次元なら、その元の高次元化対象次元が設定され、高次元化の対象でない次元なら０が設定される。１２１３，１２１６，１２１９，１２２２は各々、高次元化後の配列の１，２，３，４次元目の上下限値が設定される。

【００４２】次に、ステップ３０２でＩの値が１、Ｊの値が０となる。細分化情報１１００の１次元目の細分化フラグ１１０１の値がＴＲＵＥなので、ステップ３０３はＹｅｓとなる。次に、ステップ３０４でＪの値を１にする。ステップ３０５で高次元化辞書１２００の第１次元目の高次元化フラグ１２１１の値をｃｈａｎｇｅｄに設定する。辞書１０００中の１０４１で示された、配列ａの第１次元の下限値１と上限値１２、細分化情報１１００のフィールド１１０４が示す細分化ＰＥ数の値４より、細分化後の下限値１、細分化後の上限値１＋（１２／４）－１＝３を得る。これらから得られる下限値と上限値の組１：３を高次元化辞書１２００の第１次元目に対するフィールド１２１３に設定する。また、高次元化対応次元Ｋ＋Ｊの値３を１２１２に設定する。

【００４３】ステップ３０６で、高次元化辞書のＫ＋Ｊ

＝３次元目の高次元化フラグ１２１７の値をｎｅｗ、下限値を１、上限値を細分化ＰＥ数の値４として、組１：４を１２１９に設定する。また、高次元化対応次元Ｉの値１を１２１８に設定する。ステップ３０８で、Ｉの値が２になる。Ｉの値２は、Ｋの値２より大きいことはないので、ステップ３０９はＮｏとなり、ステップ３１０により次の次元に処理を移し、ステップ３０３に戻る。

【００４４】細分化情報１１００の２次元目の細分化フラグ１１０５の値がＴＲＵＥなので、ステップ３０３はＹｅｓとなる。ステップ３０４でＪの値は２になる。ステップ３０５で高次元化辞書１２００の第２次元目の高次元化フラグ１２１４の値をｃｈａｎｇｅｄに設定する。辞書１０００中の１０４２で示された、配列ａの第２次元の下限値１と上限値１２、細分化情報１１００のフィールド１１０８が示す細分化ＰＥ数の値４より、細分化後の下限値１、細分化後の上限値１＋（１２／４）－１＝３を得る。これらから得られる下限値と上限値の組１：３を高次元化辞書１２００の第２次元目に対するフィールド１２１６に設定する。また、高次元化対応次元Ｋ＋Ｊの値４を１２１５に設定する。

【００４５】ステップ３０６で、高次元化辞書のＫ＋Ｊ＝４次元目の高次元化フラグ１２２０の値をｎｅｗ、下限値を１、上限値を細分化ＰＥ数の値４として、組１：４を１２２２に設定する。また、高次元化対応次元Ｉの値２を１２２１に設定する。ステップ３０８で、Ｉの値が３になる。Ｉの値３は、Ｋの値２より大きいので、ステップ３０９はＹｅｓとなる。データ分散配列は他にないので、ステップ３１１はＮｏとなり、配列高次元化の処理を終了する。

【００４６】次に、プログラム変換部１０７は、辞書１２０、データ分散情報１４０及び中間語１３０を入力してプログラムを変換し、その結果を中間語１３０に反映し、またループテーブル１５０を出力するプログラム変換を行なう。プログラム変換部１０７はループ変換部１０７１と添字変換部１０７２からなる。ここでは、中間語１３０として、図８の文８０３と文８１７に対応するループｋと、文８０５から文８０９までのループｊ、ループｉのみを対象として説明し、文８１１と文８１５に対応するループｊと文８１２から文８１４までに対応するループｉに関する処理は省略する。

【００４７】図４はループ変換部１０７１の処理手順を説明したものである。プログラム分割解析ステップ４００では辞書１２０，データ分散情報１４０及び中間語１３０を入力して、ループテーブル１５０を出力する。図１３はループテーブル１５０を示す。１３００は図８の文８０３と文８１７に対応するループｋのループテーブル、１３１０は文８０５と文８０９に対応するループｊのループテーブル、１３３０は文８０６から文８０８までに対応するループｉのループテーブル、１３２０は文８１１と文８１５に対応するループｊのループテーブルを示す。文８１２から文８１４までに対応するループｉのループテーブルは説明に使わないため省略されているが、それは１３２０から接続されている。

【００４８】１３０１はループｋの一つ内側にあるループ群の先頭ループのループテーブルへのポインタであり、１３１０を指している。１３０２はループｋの一つ外側にあるループのループテーブルへのポインタであり、ループｋにはそのようなループはないので、値ＮＵＬＬが設定されている。１３０３は当該ループのループ制御変数の辞書へのポインタであり、ｋを指している。１３０４はループネストであり、一番外側のループなので、１が設定されている。１３０５は当該ループをピーリングするか否かを示すフラグであり、これはループピーリング解析４０１で設定される。１３０６はループの上下限値を示し、下限値１、上限値１０が設定されている。１３０７はループをプログラム分割した結果、作成されるプログラム分割テーブルの先頭テーブルへのポインタである。ループｋはプログラム分割対象ループでないので値ＮＵＬＬが設定されている。１３０８は後続のループのループテーブルへのポインタで、ループｋには後続ループはないので値ＮＵＬＬが設定されている。

【００４９】同様にして、１３１１はループｊの内側ループであるループｉのループテーブル１３３０へのポインタ、１３１２はループｊの外側ループであるループｋのループテーブル１３００へのポインタ、１３１３はループ制御変数ｊの辞書へのポインタ、１３１４はループｊのネスト２、１３１５はループｊをピーリングするか否かを示すフラグ、１３１６はループｊの上下限値、１３１７はプログラム分割テーブルの先頭テーブル１３４０へのポインタ、１３１８は後続ループである、８１１と８１５が示すループｊのループテーブルへのポインタを示す。

【００５０】同様にして、１３３１はループｉの内側ループがないため値ＮＵＬＬ、１３３２はループｉの外側ループであるループｊのループテーブル１３１０へのポインタ、１３３３はループｉのループ制御変数ｉの辞書へのポインタ、１３３４はループｉのネスト３、１３３５はループｉをピーリングするか否かを示すフラグ、１３３６はループｉの上下限値、１３３７はプログラム分割テーブルの先頭テーブル１３６０へのポインタ、１３３８は後続ループがないため値ＮＵＬＬを示す。ループテーブルの内、以上の内容は中間語１３０から容易に得られる。

【００５１】１３４０及び１３５０はループｊに対するプログラム分割テーブルである。１３４１は本テーブルが有効な論理プロセッサ番号の範囲を示し、論理プロセッサ番号１に対して有効であることを示す。１３４２はプログラム分割後のループの上下限値であり、下限値が２、上限値が３であることを示す。１３４３は次のプログラム分割テーブルへのポインタであり、１３５０を指

している。１３５１は本テーブルが有効な論理プロセッサ番号の範囲を示し、論理プロセッサ番号２から４までに対して有効であることを示す。１３５２はプログラム分割後のループの上下限値であり、下限値が１、上限値が３であることを示す。１３５３は次のプログラム分割テーブルへのポインタであるが、もうないので値ＮＵＬＬが設定されている。上記２つのプログラム分割テーブルにより、ループｊは論理プロセッサ番号１に対しては、ループ上下限値が２と３、論理プロセッサ番号２から４に対しては、ループ上下限値が１と３、になることがわかる。

【００５２】上記プログラム分割テーブルを得るための処理はジーマ・ヒラナンダニ、ケン・ケネディ、チャウ・ウェン・ツェン著「コンパイリング フォートラン ディーフォー エムアイエムディー ディストリビューテッド メモリ マシーンズ」コミュニケーションズ オブ ザ エーシーエム、ボリューム３５、ナンバー８、オーガスト、１９９２、第６６頁から第８０頁（Seema Hiranandani, Ken Kennedy, Chau-Wen Tseng, "Compiling Fortran D for MIMD Distributed-Memory Machines", Communications of the ACM, Vol. 35, No. 8, August, 1992, pp.66-80）に詳しい。

【００５３】次にループピーリング解析ステップ４０１は中間語１３０中の配列参照テーブルとループテーブルを入力し、各々に対してループピーリング後の情報を追加する。

【００５４】図１４は文８０７中のａ（ｉ，ｊ－１）に対する配列参照テーブルを示したものである。１４００はａ（ｉ，ｊ－１）に対する配列参照テーブルである。１４０１は配列ａの辞書へのポインタである。１４０２は配列の各次元の添字情報を表わす添字テーブルの先頭へのポインタであり、１４１０を指している。１４０３は各次元ごとのピーリング情報を表わすテーブルへのポインタであり、１４３０を指している。これはループピーリング解析ステップ４０１で設定される。１４１０はａ（ｉ，ｊ－１）の１次元目の添字テーブルである。１４１１は２次元目に対する添字テーブルへのポインタであり、１４２０を指している。１４１２は添字を表わす中間語へのポインタであり、ｉを表わす中間語を指している。１４２０はａ（ｉ，ｊ－１）の２次元目の添字テーブルである。１４２１は配列ａには３次元目が存在しないので、値ＮＵＬＬが設定されている。１４２２は添字を表わす中間語へのポインタであり、ｊ－１を表わす中間語を指している。１４３０及び１４４０は次のループピーリング解析で作成されるテーブルであり、その処理の中で説明する。

【００５５】図５はループピーリング解析ステップ４０１の詳細な処理手順を示したものである。ここでは図１４で示された配列参照テーブル１４００に対する処理のみ説明する。ステップ５００でＩの値は１になる。配列ａの１次元目は細分化情報１１００中のフィールド１１０１がＴＲＵＥなので細分化する次元であり、ステップ５０１はＹｅｓとなる。

【００５６】１４１２より当次元に現れる変数はｉであり、これはループテーブル１３３０よりループ制御変数であることがわかる。このｉに分散後ループ範囲１３６２を代入して、当次元の添字範囲は１：３になる。一方、データ分散後の当次元の添字範囲は高次元化辞書１２００のフィールド１２１３より１：３であるから、分散後ループ範囲から得られた添字範囲１：３からデータ分散後の添字範囲１：３をひいた残りの範囲は空集合となり、ステップ５０２の結果は空集合である。よって、ステップ５０３はＹｅｓとなる。

【００５７】ステップ５０５で当次元はピーリングしない、とし、ピーリングすべきループ範囲とピーリングすべき添字のはみ出す方向を各々、０とする。次に、ステップ５０６で、ピーリング情報テーブル１４３０を作成する。ここで、１４３１は次のピーリング情報テーブルへのポインタ，１４３２はピーリングするか否かを示すフラグ，１４３３は当次元に現れるループ制御変数，１４３４はピーリングすべきループ範囲，１４３５はピーリングすべき添字のはみ出す方向，１４３６は当ループ制御変数を持つループテーブルへのポインタである。

【００５８】よって、今の場合、１４３１には、まずＮＵＬＬが設定される。但し、このフィールドは次の２次元目の処理で、新規に作成されるピーリング情報テーブル１４４０を指すように変更される。１４３２にはピーリングしないのでＦＡＬＳＥが、１４３３には当次元に現れるループ制御変数ｉが、１４３４には０が、１４３５にも０が、１４３６にはループｉに対するループテーブル１３３０へのポインタが設定される。

【００５９】次に、ステップ５０７では、現在は１次元目の処理をしているので、Ｙｅｓとなる。ステップ５０８で、Ｉの値は２になる。配列ａの２次元目は細分化情報１１００中のフィールド１１０５がＴＲＵＥなので細分化する次元であり、ステップ５０１はＹｅｓとなる。１４２２より当次元に現れる変数はｊであり、これはループテーブル１３１０よりループ制御変数であることがわかる。このｊに分散後ループ範囲１３４２、１３５２を代入し、１３４１、１３５１の論理プロセッサ番号範囲も合わせると、当次元の添字範囲は、論理プロセッサ番号範囲が１：１の時は１：２、論理プロセッサ番号範囲が２：４の時は０：２となる。

【００６０】一方、データ分散後の当次元の添字範囲は高次元化辞書１２００のフィールド１２１６より１：３であるから、分散後ループ範囲から得られた添字範囲からデータ分散後の添字範囲をひいた残りの範囲は論理プロセッサ番号範囲が２：４の時に０：０となり、ステップ５０２の結果は０：０である。よって、ステップ５０３はＮｏとなる。

【００６１】ステップ５０４では、まず、「当次元はピーリングする」とする。次に、０：０が２次元目の添字であるｊ－１と等しくなることより、ｊの値は１：１となる。よって、はみ出す時のループ範囲であるピーリング範囲は１：１である。また、このピーリング範囲は論理プロセッサ番号範囲が２：４の時に生じるので、ループテーブル１３１０の分散後ループ範囲のうち、この論理プロセッサ番号範囲に対応するテーブル１３５０を細分する。即ち、テーブル１３５０を、ピーリング範囲１：１の範囲を持つテーブルとそれ以外の範囲からなるテーブルに分割する。

【００６２】図１５は細分化後のループ分散範囲テーブルである。テーブル１３５０のループ分散範囲は１：３から１：１に変更されている。新たにテーブル１５００が作成され、１３５３は１５００を指す。１５０１は細分されるループ分散範囲の論理プロセッサ番号範囲２：４に設定される。１５０２は元のループ分散範囲は１：３から１：１を引いた範囲２：３に設定される。１５０３は他に指すべきテーブルがないので、値ＮＵＬＬが設定される。

【００６３】次に、ステップ５０６より、ピーリング情報テーブル１４４０を作成する。まず、１４４１には次のテーブルがないのでＮＵＬＬが設定され、１４３１には１４４０へのポインタが設定される。１４４２には、ピーリングすることを示すフラグＴＲＵＥが、１４４３には当次元に現れるループ制御変数であるｊが、１４４４にはステップ５０４で計算した１：１を設定する。ループ分散後のループ範囲１３４２及び１３５２の内、１：１は１３５２の下限を含むので、はみ出す方向として下限方向を示す－１を、１４４５に設定する。１４４６には、ループｊに対するループテーブル１３１０へのポインタを設定する。

【００６４】次に、配列ａにはもう処理すべき次元がないので、ステップ５０７はＮｏとなる。上記説明したように、図５では図１４の配列参照に対する処理だけを説明するので、ステップ５０９はＮｏになり、これでループピーリング処理の説明を終了する。

【００６５】ここで、図４に戻り、ループ変換の説明を続ける。ループ変換の残りの処理では、図１３のループテーブル群から図１６及び１７で示されたループテーブル群を生成し、図１８のプログラムを生成する。まず、ステップ４０２で、最初の最内側ループであるループｉを検出する。以降の処理で、処理済みの分散ループに対しては、ループ分散範囲テーブルへのポインタをＮＵＬＬにする。よって、ステップ４０３で、未処理の分散ループであることは、分散ループであることを示すフラグがＴＲＵＥで、かつ、ループ分散範囲テーブルへのポインタがＮＵＬＬでないことからわかる。図１３の、ループｉに対するループテーブル１３３０で、１３３５はＴＲＵＥで、１３３７はＮＵＬＬでないので、ループｉは未処理の分散ループであり、ステップ４０３はＹｅｓである。

【００６６】ステップ４０４では、１３３０の一つ内側にループテーブルを作成する。図１６はステップ４０４により現ループテーブル１３３０（ＬＴ１）の一つ内側に新たにループテーブル１６００（ＬＴ２）が作成された様子を表わした図である。

【００６７】以下、ステップ４０４に従って図１６を説明する。まず、ループ分散テーブル１３６０は１個だけなので、ＬＴ１以下のループテーブルのコピーは作成する必要はない。ＬＴ１に対するループ制御変数名をｉ２とし、１３３３に設定する。ＬＴ１のループ範囲１３３６をループ分散範囲テーブル１３６０の論理プロセッサ番号範囲１３６１である１：４に設定する。

【００６８】ループテーブル１６００は一つ内側のループに対するテーブルなので、一つ内側のループ先頭を指すポインタ１３３１を１６００を指すように設定する。一つ外側のループテーブルへのポインタ１３３１，ループネスト１３３４，分散ループフラグ１３３５，後続ループへのポインタ１３３８はそのままである。ループ分散範囲テーブル１３６０へのポインタ１３３７をＮＵＬＬに設定して、このループテーブルが処理済みであることを示す。

【００６９】次に、ＬＴ２に対するループ制御変数名をｉ１とし、１６０３に設定する。ＬＴ２のループ範囲１６０６をループ分散範囲テーブル１３６０のループ分散範囲１３６２である１：３に設定する。ループテーブル１６００は最内側ループなので、１６０１はＮＵＬＬに、一つ外側のループテーブルへのポインタである１６０２は１３３０を指すように設定される。また、１６０４，１６０５，１６０７，１６０８は各々、０、ＴＲＵＥ、ＮＵＬＬ、ＮＵＬＬに初期化される。１６０５がＴＲＵＥで１６０７がＮＵＬＬなので、ループテーブル１６００は処理済みであることを示す。

【００７０】次に、１３３８はＮＵＬＬなので後続ループはなく、ステップ４０５はＮｏである。１３３２は外側ループテーブル１３１０を指しているので、ステップ４０７はＴＲＵＥ（Ｙｅｓ）であり、ステップ４０８で処理は外側ループテーブル１３１０に移り、ステップ４０３へ戻る。１３１５はＴＲＵＥだが、１３１７はＮＵＬＬでないので、ループテーブル１３１０は未処理の分散ループであり、ステップ４０３はＹｅｓである。

【００７１】ループテーブル１３１０のループ分散範囲テーブルの個数は図１５より３個なので、ステップ４０４では、ループテーブル１３１０（ＬＴ１）の一つ内側にループテーブル（ＬＴ２）を作成し、ＬＴ１以下のループテーブルのコピーを２個作成して、ＬＴ１の後続ループとして接続する。

【００７２】図１７はこれらのテーブルが作成された様子を表わした図である。１７００がＬＴ２を、１７１

０，１７２０，１７３０，１７４０がＬＴ１以下のループテーブルの第１のコピーを、１７５０，１７６０，１７７０，１７８０がＬＴ１以下のループテーブルの第２のコピーを表わす。

【００７３】以下、ステップ４０４に従って図１７を説明する。まず、ＬＴ１とそのコピーに対するループ制御変数名をｊ２とし、１３１３，１７１３，１７５３に設定する。ＬＴ１とそのコピーのループ範囲１３１６，１７１６，１７５６をループ分散範囲テーブル１３４０，１３５０，１５００の論理プロセッサ番号範囲１３４１，１３５１，１５０１である１：１，２：４，２：４に設定する。ループテーブル１７００，１７２０，１７６０は、各々、ループテーブル１３１０，１７１０，１７５０の一つ内側のループに対するテーブルなので、一つ内側のループ先頭を指すポインタ１３１１，１７１１，１７５１を、各々、１７００，１７２０，１７６０を指すように設定する。一つ外側のループテーブルへのポインタ１３１２はそのままであり、１７１２，１７５２はＮＵＬＬに初期化される。

【００７４】ループネスト１３１４はそのままであり、１７１４，１７５４は０に初期化される。これらの値は後のループネスト決定ステップ４０９で、括弧内の値に設定される。分散ループフラグ１３１５はそのままであり、１７１５，１７５５はＴＲＵＥに設定される。ループ分散範囲テーブル１３４０へのポインタ１３１７、及び１７１７，１７５７をＮＵＬＬに設定して、これらのループテーブルが処理済みであることを示す。後続ループテーブルへのポインタは、１３２０を指していた１３１８は１７１０を、１７１８は１７５０を、１７５８は１３２０を指すように設定される。

【００７５】次に、ＬＴ２とそのコピーに対するループ制御変数名をｊ１とし、１７０３，１７２３，１７６３に設定する。ＬＴ２とそのコピーのループ範囲１７０６，１７２６，１７６６をループ分散範囲テーブル１３４０，１３５０，１５００のループ分散範囲１３４２，１３５２，１５０２である２：３，１：１，２：３に設定する。

【００７６】ループテーブル１３３０，１７３０，１７７０は、各々、ループテーブル１７００，１７２０，１７６０の一つ内側のループに対するテーブルなので、一つ内側のループ先頭を指すポインタ１７０１，１７２１，１７６１を、各々、１３３０，１７３０，１７７０を指すように設定する。ループテーブル１３１０，１７１０，１７５０は、各々、ループテーブル１７００，１７２０，１７６０の一つ外側のループに対するテーブルなので、一つ外側のループ先頭を指すポインタ１７０２，１７２２，１７６２を、各々、１３１０，１７１０，１７５０を指すように設定する。ループネスト１７０４，１７２４，１７６４は０に初期化される。これらの値は後のループネスト決定４０９で、括弧内の値に設定される。分散ループフラグ１７０５，１７２５，１７６５はＴＲＵＥに設定される。ループ分散範囲テーブルへのポインタ１７０７，１７２７，１７６７をＮＵＬＬに設定して、こちらのループテーブルが処理済みであることを示す。後続ループテーブルへのポインタは１７０８，１７２８，１７６８はＮＵＬＬに初期化される。

【００７７】１７３０と１７７０、１７４０と１７８０は各々、１３３０，１６００のコピーであり、外側ループ、内側ループを指すポインタ、ループネスト以外は全て同じ値となる。即ち、１７３３，１７７３とは１３３３と同じ値で、１７３５から１７３８まで、１７７５から１７７８までは１３３５から１３３８までと同じ値である。また、１７４３，１７８３は１６０３と同じ値で、１７４５から１７４８までと１７８５から１７８８までは１６０５から１６０８までと同じ値である。ループネストである１７３４，１７７４，１７４４，１７８４は０に初期化される。一つ内側のループテーブルを指すポインタは、１７３１が１７４０を、１７４１はＮＵＬＬを、１７７１は１７８０を、１７８１はＮＵＬＬを指すように設定される。一つ外側のループテーブルを指すポインタは、１７３２が１７２０を、１７４２は１７３０を、１７７２は１７６０を、１７８２は１７７０を指すように設定される。

【００７８】次に、ループテーブル１３１０の元々の後続テーブルは１３２０であるが、本実施例では１３２０以降のテーブルに対する処理は説明しないので、ステップ４０５はＮｏとなる。１３１２より、ループテーブル１３１０の外側ループテーブル１３００が存在するので、ステップ４０７はＹｅｓとなり、ステップ４０８で処理をループテーブル１３００に移し、ステップ４０３へ戻る。

【００７９】１３０５がＦＡＬＳＥなので、ループテーブル１３００に対応するループｋは分散ループでなく、ステップ４０３はＮｏとなる。１３０８はＮＵＬＬなので、後続ループはなく、ステップ４０５はＮｏとなる。１３０２はＮＵＬＬなので、外側ループはなく、ステップ４０７はＮｏとなる。

【００８０】次に、ループネスト決定ステップ４０９で、図１７のループテーブルのネストを決定する。この処理は最も外側のループから順番に決定する簡単な処理なので、詳細は省略する。図１７の、１３０４，１３１４，及び，１７０４，１３３４，１６０４，１７１４，１７２４，１７３４，１７４４，１７５４，１７６４，１７７４，１７８４の括弧内の数値が決定されたループネストである。

【００８１】次にループ多重化・ループピーリングステップ４１０で図１７のループテーブル群に従って、ループを表わす元の中間語に、ループ多重化とループピーリングを適用する。図１８は、元のループ中間語である図８の文８０３から文８１７に、ループ多重化とループピ

ーリングを適用した結果のプログラムである。

【００８２】文１８１０から文１８３５までが図８の文８０５から文８０９に対応する部分である。このうち、文１８１０から文１８１５までがループテーブル１３１０，１７００，１３３０，１６００に対応し、文１８２０から文１８２５までがループテーブル１７１０，１７２０，１７３０，１７４０に対応し、文１８３０から文１８３５までがループテーブル１７５０，１７６０，１７７０，１７８０に対応する。同様にして、文１８４０から文１８８０までが図８の文８１１から文８１５に対応する部分である。

【００８３】図４に戻り、最後に後続ループネストはないので、ステップ４１１はＮｏとなり、ループ変換部１０７１の処理を終了する。

【００８４】次に、図１の添字変更部１０７２の処理を行なう。図６は添字変更部１０７２の処理手順を説明した図である。ここでは、文１８１４と文１８２４中の配列参照ａ（ｉ，ｊ－１）に対する処理を、その配列参照テーブルを示した図１４、処理結果得られるプログラムを示した図１９を用いて説明する。

【００８５】まず、ステップ６００でＩの値は１となり、１次元目の処理を行なう。細分化情報テーブル中の１次元目の細分化フラグ１１０１の値がＴＲＵＥなので、この次元は細分化する次元であり、ステップ６０１はＹｅｓとなる。高次元化辞書１２００の１次元目に対する高次元化対応次元１２１２は３となっているので、ステップ６０２では、配列参照に３次元目を追加する。文１８１４はループ中にあるので、ステップ６０３はＹｅｓである。

【００８６】ステップ６０４で、現在の配列参照テーブル１４００の１次元目の添字は１４１１よりｉで、ループ制御変数ｉを持っていたループテーブルへのポインタは１４３６より１３３０とわかるので、１次元目の添字を値が先に増加する内側ループのループ制御変数である１６０３のｉ１に、３次元目の添字を外側ループのループ制御変数である１３３３のｉ２に設定する。現配列参照の１次元目に対するピーリングフラグ１４３２はＦＡＬＳＥなので、ステップ６０６はＮｏとなる。配列ａは元々２次元なので、ステップ６０８はＹｅｓとなり、ステップ６１０でＩの値を２にして処理を２次元目に移し、ステップ６０１に戻る。

【００８７】細分化情報テーブル中の２次元目の細分化フラグ１１０５の値がＴＲＵＥなので、この次元は細分化する次元であり、ステップ６０１はＹｅｓとなる。高次元化辞書１２１０の２次元目に対する高次元化対応次元１２１５は４となっているので、ステップ６０２では、配列参照に４次元目を追加する。文１８１４はループ中にあるので、ステップ６０３はＹｅｓである。

【００８８】ステップ６０４で、現在の配列参照テーブル１４００の２次元目の添字は１４２２よりｊ－１で、ループ制御変数ｊを持っていたループテーブルへのポインタは１４４６より１３１０とわかるので、２次元目の添字を値が先に増加する内側ループのループ制御変数である１７０３のｊ１を用いてｊ１－１に、４次元目の添字を外側ループのループ制御変数である１３１３のｊ２に設定する。

【００８９】現配列参照の２次元目に対するピーリングフラグ１４４２はＴＲＵＥだが、現配列参照のピーリングすべきループ範囲は１４４４より１：１である。一方、現配列参照を含み、元々ピーリングループ制御変数ｊを持っていたループのループテーブル１３１０の、一つ内側ループのループ範囲１７０６は２：３であるから、１：１は２：３に含まれておらず、ステップ６０６はＮｏとなる。これは、ピーリングに伴って、ｊに対する添字変更の特別処理が必要なループ範囲が１：１であるが、現在のループ範囲はその範囲を含んでいないので、特別な処理が不要であることを示している。

【００９０】配列ａの次元はもうないので、ステップ６０８はＮｏとなる。以上で、文１８１４中の配列参照ａ（ｉ，ｊ－１）は、文１９１５中の配列参照ａ（ｉ１，ｊ１－１，ｉ２，ｊ２）に変更される。配列参照はまだあるので、ステップ６０９はＹｅｓとなり、ステップ６１１で次の配列参照である文１８２４のａ（ｉ，ｊ－１）に処理を移す。

【００９１】この配列参照に対して、１次元目の処理は文１８１４の場合と同じなので、省略して、２次元目の場合のみ説明する。ステップ６０１，６０２，６０３，６０４の処理は文１８１４の配列参照の場合と同様で、その結果、文１８２４のａ（ｉ，ｊ－１）は、一旦、文１９１５の配列参照ａ（ｉ１，ｊ１－１，ｉ２，ｊ２）のように変更される。

【００９２】次に、ピーリングフラグ１４４２はＴＲＵＥで、現配列参照のピーリングすべきループ範囲は１４４４より１：１である。一方、現配列参照を含み、元々ピーリングループ制御変数ｊを持っていたループのループテーブル１３１０のコピーになるループテーブルは１７１０であり、その一つ内側ループのループ範囲１７２６は１：１である。よって、両者は一致し、ステップ６０６はＹｅｓとなる。ピーリング添字のはみ出す方向は１４４５より－１、現次元の寸法は１２１６より３なので、ｊ１はｊ１＋３，ｊ２はｊ２－１となる。これより、ｊ１－１はｊ１＋２に、ｊ２はｊ２－１になり、文１９２５の配列参照ａ（ｉ１，ｊ１＋２，ｉ２，ｊ２－１）が得られる。

【００９３】次に、配列ａは２次元配列なので、ステップ６０８はＮｏとなる。本説明では２つの配列参照についてのみ説明するので、ステップ６０９はＮｏとなり、添字変更部１０７２の処理を終了する。他の配列参照についても同様に処理することで、図１９の中間語が得られる。以上により、プログラム変換部１０７の処理を終

了する。

【００９４】次に、プログラム並列化部１０８の処理を説明する。図７はプログラム並列化部１０８の処理手順を説明した図である。データ分散情報変更ステップ７００は、データ分散情報１４０と高次元化辞書１２００を入力して、変更したデータ分散情報を出力する。図２０は変更後のデータ分散情報である。データ分散情報変更の処理は容易なので以下、概略を記す。

【００９５】配列高次元化部１０６の処理によって、配列の次元が１２０１に示されるように４になったので、図２０における９０２，９１２は図９における２から４になる。これに伴い、２０００，２０１０には４次元分のテーブルを確保する。次に、高次元化辞書において、フラグの値が“ｃｈａｎｇｅｄ”で、かつ、データ分散が指定された次元の分散形状は、次元毎分散情報テーブルの該当する次元では“＊”と変更され、上記次元に対応する、フラグの値が“ｎｅｗ”である次元の分散形状は、上記“ｃｈａｎｇｅｄ”フラグを持つ次元の、元の分散形状に設定される。但し、ブロック分割の幅は再計算される。その他の次元は、“＊”と設定される。

【００９６】まず、図９のデータ分散情報テーブル９００について処理する。高次元化辞書の第１次元目は、フラグ１２１１の値が“ｃｈａｎｇｅｄ”で、データ分散情報テーブル９００に対応する第１次元目のデータ分散形状９２１はｂｌｏｃｋなので、この次元の分散形状は“＊”に変更される。２００１は“＊”となり、２００２には１２１３より、高次元化辞書の第１次元目の寸法３が設定される。上記第１次元目に対応する次元は１２１２より３次元目であり、３次元目のフラグ１２１７の値が“ｎｅｗ”なので、この次元はデータ分散するように変更される。よって、２００５はｂｌｏｃｋとなり、この次元の上下限１２１９よりわかる寸法４を、９５０よりわかる論理プロセッサ数４で割った値１がブロック分割の幅となるので、この値１を２００６に設定する。

【００９７】一方、第２次元目は、フラグ１２１４の値は“ｃｈａｎｇｅｄ”であるが、データ分散情報テーブル９００に対応する第２次元目のデータ分散形状９２３は“＊”なので、この次元の分散形状は“＊”に設定される。よって、２００３は“＊”となり、２００４には１２１６より、高次元化辞書の第２次元目の寸法３が設定される。また、上記２次元目に対応する次元は１２１５より４となり、第４次元目の分散形状も“＊”に設定される。よって、２００７は“＊”となり、２００８には１２２２より、高次元化辞書の第４次元目の寸法４が設定される。

【００９８】次に、図９のデータ分散情報テーブル９１０について処理する。高次元化辞書の第１次元目は、フラグ１２１１の値は“ｃｈａｎｇｅｄ”であるが、データ分散情報テーブル９１０に対応する第１次元目のデータ分散形状９３１は“＊”なので、この次元はデータ分散しないように設定される。よって、２０１１は“＊”となり、２０１２には１２１３より、高次元化辞書の第１次元目の寸法３が設定される。また、上記第１次元目に対応する次元は１２１２より３となり、第３次元目もデータ分散されないように設定される。よって、２０１５は“＊”となり、２０１６には１２１９より、高次元化辞書の第３次元目の寸法４が設定される。

【００９９】一方、高次元化辞書の第２次元目は、フラグ１２１０の値が“ｃｈａｎｇｅｄ”で、データ分散情報テーブル９１０に対応する第２次元目のデータ分散形状９３３はｂｌｏｃｋなので、この次元はデータ分散しないように変更される。２０１３は“＊”となり、２０１４には１２１６より、高次元化辞書の第２次元目の寸法３が設定される。上記第２次元目に対応する次元は１２１５より４次元目であり、４次元目のフラグ１２２０の値が“ｎｅｗ”なので、この次元はデータ分散するように変更される。よって、２０１７はｂｌｏｃｋとなり、この次元の上下限１２２２よりわかる寸法４を、９５０よりわかる論理プロセッサ数４で割った値１がブロック分割の幅となるので、この値１を２０１８に設定する。

【０１００】次に、図２０のデータ分散情報に基づいて、プログラム分割解析ステップ７０１、通信解析ステップ７０２を実行する。これらに対する処理方法はジーマ・ヒラナンダニ、ケン・ケネディ、チャウ・ウェン・ツェン著「コンパイリング フォートラン ディー フォー エムアイエムディー ディストリビューテッド メモリ マシーンズ」コミュニケーションズ オブ ザ エーシーエム、ボリューム３５、ナンバー８、オーガスト、１９９２、第６６頁から第８０頁（Seema Hiranandani, Ken Kennedy, Chau-Wen Tseng, "Compiling Fortran D for MIMD Distributed-Memory Machines", Communications of the ACM, Vol. 35, No. 8, August,1992, pp.66-80）に書かれている。また、通信生成７０７、プログラム分割コード生成７０８、分割データ宣言作成７０９についても同文献に書かれている。

【０１０１】図２１はプログラム並列化の処理結果のプログラムを示す図である。以下、図１０，図２０，図２１を用い、処理手順の概略を説明する。配列ａは細分化フラグ１００５がＴＲＵＥなので、ステップ７０３はＹｅｓとなる。通信生成ステップ７０４では、通常のｓｅｎｄ／ｒｅｃｅｉｖｅ型の論理プロセッサ間通信の生成を行い、再分散処理ではステップ７０４１の非バッファリング再分散通信生成により、バッファリングしない再分散通信を生成する。これは図２１の文２１０２及び文２１４０に対応する。２１０１の再分散ルーチン“ｎｏｎ＿ｂｕｆ＿ｒｅｍａｐ”は、配列ａを、ディスクリプタＤ１に記述されたデータ分散形状から、ディスクリプタＤ２に記述されたデータ分散形状に、バッファを使わずにデータ再分散するルーチンである。

【０１０２】文２１０２のＤ１は、この時点における配列ａのデータ分散形状を記述したディスクリプタである。これは、ループｋの１回目の繰り返しでは、プログラム先頭におけるデータ分散であり、文８０２と文８０４が元々同じデータ分散であることから、文８０４でのデータ分散に対応する、図２０のテーブル９００及び２０００が示すデータ分散となる。また、ループｋの２回目以降の繰り返しでは、文８１０でのデータ分散に対応する、図２０のテーブル９１０及び２０１０が示すデータ分散となる。

【０１０３】文２１０２のＤ２は、再分散ルーチン“ｎｏｎ＿ｂｕｆ＿ｒｅｍａｐ”により、配列ａが再分散された後のデータ分散形状を記述したディスクリプタであり、図２０のテーブル９００及び２０００が示すデータ分散情報に対応している。文２１４０の再分散ルーチン“ｎｏｎ＿ｂｕｆ＿ｒｅｍａｐ”は、配列ａを、ディスクリプタＤ２に記述されたデータ分散形状から、ディスクリプタＤ１に記述されたデータ分散形状に、バッファを使わずにデータ再分散するルーチンである。

【０１０４】文２１４０のＤ２は、この時点における配列ａのデータ分散形状を記述したディスクリプタであり、図２０のテーブル９００及び２０００が示すデータ分散情報に対応している。文２１４０のＤ１は、この時点における配列ａのデータ分散形状を記述したディスクリプタであり、図２０のテーブル９１０及び２０１０が示すデータ分散情報に対応している。

【０１０５】プログラム分割コード生成ステップ７０５では、ループインデックス生成ステップ７０５１とグローバル添字生成ステップ７０５２の処理を実行する。ループインデックス生成ステップ７０５１では、データ分散する配列次元の添字に含まれるループ制御変数に対応するループのループ範囲を分散して、分散後のループの上下限値を生成する。

【０１０６】本発明の配列高次元化部１０６の処理を適用した後の配列では、データ分散情報変更ステップ７００によりデータ分散する次元は元のデータ分散次元の高次元化対応次元であり、その次元の寸法は配列高次元化部１０６の処理より論理プロセッサ数と等しい。ループ変換部１０７１により上記データ分散される次元に現れるループ制御変数を持つループのループ長はループ分散範囲テーブルの論理プロセッサ番号範囲に等しく、図８に対する中間語を入力とした場合、それは論理プロセッサ数に等しいのでループは解消される。

【０１０７】グローバル添字生成ステップ７０５２では、データ分散した次元の添字として、データ分散前の該当する配列要素を表わす添字を用いる。本発明では、データ分散する配列次元の添字はデータ分散前は、論理プロセッサ番号の内のいずれかと等しくなっているので、データ分散後の添字も同じ論理プロセッサ番号にする。

【０１０８】文１９１０から文１９３６までのデータ分散は、データ分散情報テーブル９００と２０００で示されたものであり、配列ａの第３次元目がブロック分割される。ａの第３次元目にあるループ制御変数は文１９１４，１９１５，１９２４，１９２５，１９３４，１９３５の配列参照においてｉ２であるので、ループｉ２は解消され、配列ａの第３次元目の添字は論理プロセッサ番号になる。図２１の文２１１３，２１１４，２１２３，２１２４，２１３３，２１３４における配列ａの第３次元目の添字“ｍｙｐｅ”が論理プロセッサ番号を表わしており、図２１の文２１１０から２１３５には、ループｉ２がないことがわかる。

【０１０９】同様にして、文１９４０から文１９８０までのデータ分散は、データ分散情報テーブル９１０と２０１０で示されたものであり、配列ａの第４次元目がブロック分割される。ａの第４次元目にあるループ制御変数は文１９５２，１９５３，１９６２，１９６３，１９７２，１９７３の配列参照においてｊ２であるので、ループｊ２は解消され、配列ａの第４次元目の添字は論理プロセッサ番号になる。図２１の文２１５２，２１５３，２１６２，２１６３，２１７２，２１７３における配列ａの第４次元目の添字“ｍｙｐｅ”が論理プロセッサ番号を表わしており、図２１の文２１４１から文２１８０には、ループｊ２がないことがわかる。

【０１１０】次にグローバルデータ宣言作成ステップ７０６では、データ分散配列に対しても、高次元化後で、データ分散前の配列宣言を行なう。文２１００の宣言がそれに該当する。最後に、コード生成部１０９は、図２１に対応する中間語を入力し、図２１のソースプログラムまたはそれをオブジェクトプログラムの形にした並列化プログラム１６０を出力する。これで第１の実施例の処理フローの説明を終了する。

【０１１１】図２２は本願のデータ再配置やデータ分散を適用する前の配列を用いて、データ再分散の様子を説明したものである。２２００は１時限目および２次元目の要素数が各々１２の２次元配列ａを表わし、縦方向は配列の１次元目を表わす。この次元はＦＯＲＴＲＡＮではアドレスの連続する次元にあたる。また、横方向は配列の２次元目を表わす。２２００中の１６個の矩形は各々、３ｘ３の部分配列を表わす。

【０１１２】各部分配列中の矢印の左側の数字は、配列ａが（ｂｌｏｃｋ，＊）という方法で４台の論理プロセッサにブロック分散された時の、各々の部分配列が割り付けられる論理プロセッサ番号を、矢印の右側の数字は、配列ａが（＊，ｂｌｏｃｋ）という方法で４台の論理プロセッサにブロック分散された時の、各々の部分配列が割り付けられる論理プロセッサ番号を表わす。したがって、各部分配列中の矢印とその両側の数字により、その部分配列が（ｂｌｏｃｋ，＊）から（＊，ｂｌｏｃｋ）へのデータ再分散によって、左側の番号の論理プロ

セッサから右側の番号の論理プロセッサへ送信されることを表わす。

【０１１３】図２３は従来技術による並列化プログラムである。文２３００における配列ａの宣言が示すように、配列は第１次元目が分割されて寸法が３になっている。文２３０２と文２３２０はバッファリングをするデータ再分散ルーチンである。文２３０２の再分散ルーチン“ｒｅｍａｐ”は、配列ａを、ディスクリプタＥ１に記述されたデータ分散形状から、ディスクリプタＥ２に記述されたデータ分散形状に、データ再分散するルーチンである。

【０１１４】文２３０２のＥ１は、この時点における配列ａのデータ分散形状を記述したディスクリプタである。これは、ループｋの１回目の繰り返しでは、プログラム先頭におけるデータ分散形状であり、文８０２と文８０４が同じデータ分散であることから、文８０４のデータ分散を表わす、図９のテーブル９００及び９２０が示すデータ分散となる。また、ループｋの２回目以降の繰り返しでは、文８１０でのデータ分散を表わす、図９のテーブル９１０及び９３０が示すデータ分散となる。

【０１１５】文２３０２のＥ２は、再分散ルーチン“ｒｅｍａｐ”により、配列ａが再分散された後のデータ分散形状を記述したディスクリプタであり、図９のテーブル９００及び９２０が示すデータ分散情報に対応している。文２３２０の再分散ルーチン“ｒｅｍａｐ”は、配列ａを、ディスクリプタＥ２に記述されたデータ分散形状から、ディスクリプタＥ１に記述されたデータ分散形状に、データ再分散するルーチンである。文２３２０のＥ２は、この時点における配列ａのデータ分散形状を記述したディスクリプタであり、図９のテーブル９００及び９２０が示すデータ分散情報に対応している。文２３２０のＥ１は、再分散ルーチン“ｒｅｍａｐ”により、配列ａが再分散された後のデータ分散形状を記述したディスクリプタであり、図９のテーブル９１０及び９３０が示すデータ分散情報に対応している。

【０１１６】図２４は従来の並列化によるデータ再分散の様子を、特に、論理プロセッサ１（以下、Ｐ１と略す）から論理プロセッサ２（以下、Ｐ２と略す）へ送信されるデータに注目して表示したものである。２４０１で示される３行１２列の部分配列は、配列ａを（ｂｌｏｃｋ，＊）というデータ分散方法で従来の並列化方法により並列化した時、Ｐ１に割り付けられる分割データである。２４００は分割データ２４０１の元の配列における相対的な位置を示すために、参考までに表示した１２行１２列の元の配列である。配列ａは（ｂｌｏｃｋ，＊）でデータ分散されるため、配列の１次元目が４分割され、その内の一つである２４０１がＰ１に割り付けられるデータとなる。２４０１中の数字は、Ｐ１に割り付けられたデータ２４０１の配列要素がアドレス順に並ぶ様子を示している。

【０１１７】２４１１で示される１２行３列の部分配列は、配列ａを（＊，ｂｌｏｃｋ）というデータ分散方法で従来の並列化方法により並列化した時、Ｐ２に割り付けられる分割データである。２４１０は分割データ２４１１の元の配列における相対的な位置を示すために、参考までに表示した１２行１２列の元の配列である。配列ａは（＊，ｂｌｏｃｋ）でデータ分散されるため、配列の２次元目が４分割され、その内の一つである２４１１がＰ２に割り付けられるデータとなる。２４１１中の数字は、Ｐ２に割り付けられたデータ２４１１の配列要素がアドレス順に並ぶ様子を示している。

【０１１８】２４０１中の網掛け部分２４０２は、（ｂｌｏｃｋ，＊）から（＊，ｂｌｏｃｋ）へのデータ再分散により、Ｐ１からＰ２へ送信されるデータを表わす。図中に示されたアドレス順により、このデータは１９番目から２７番目までの配列要素からなる連続データであることがわかる。一方、２４１１中の網掛け部分２４１２は、（ｂｌｏｃｋ，＊）から（＊，ｂｌｏｃｋ）へのデータ再分散により、Ｐ２がＰ１から受信するデータの格納先を表わす。図中に示されたアドレス順により、このデータは４番目から６番目，１６番目から１８番目，２８番目から３０番目までの配列要素からなる非連続データであることがわかる。

【０１１９】従って、送信側データは連続であるが、受信側では非連続な位置にデータを格納することが必要であるとわかる。データ通信ルーチンの中には、送信側データは非連続なものを許すものもあるが、受信側は通常、連続になってないといけないので、この場合、１回のデータ送信で済ます場合には、受信側でどうしてもバッファリングが必要なことがわかる。即ち、送信側データ２４０２は送信ルーチンにより、Ｐ２内のアドレスが連続するバッファに送信され、受信したＰ２は、そのバッファ内のデータを２４１２へ、各列毎に１回、少なくとも計３回コピーを行なう。

【０１２０】図２５は、本発明の並列化によるデータ再分散の様子を、Ｐ１からＰ２へ送信されるデータに注目して表示したものである。２５００で示される１２行１２列の配列は、配列ａを（ｂｌｏｃｋ，＊）というデータ分散方法で分割して、本発明の並列化方法により並列化した時、Ｐ１に割り付けられるデータである。この配列の大きさはデータ分散前の配列の大きさと同じである。２５００中の数字は、Ｐ１に割り付けられたデータ配列２５００の配列要素がアドレス順に並ぶ様子を示している。左上角から始まる１２行３列の部分に、最初の３６要素のアドレス順が示されている。これにより、太線で区切られた３行３列の部分は連続アドレスであることがわかる。数字が表示されてない他の部分についても同様である。

【０１２１】この、本発明による配列要素のアドレスの変更は、一般に、配列ａを２次元配列ａ（Ｎ，Ｎ）、プ

ロセッサ数をＰ、ＮはＰで割り切れる、とする時、次の＊　＊式（１）によって表現される。

(i,j)→((i-1)modL+1, (j-1)modL+1, [(i-1)/L]+1, [(J-1)/L]+1)　　（１）

ここで、矢印の左辺の（ｉ，ｊ）は元の配列ａの第ｉ行第ｊ列の配列要素を表し、矢印の右辺は本発明による配列再配置を適用した後の４次元配列における対応する配列要素を表わす。また、ＬはＮ／Ｐを、［］は切り捨てを表わすガウスの記号である。

【０１２２】この４次元配列の最初の２次元から作られる２次元配列は、データ配列２５００中の、太線で区切※られた３行３列の部分に対応し、最後の２次元から作られる２次元配列は、この太線で区切られた３行３列の部分を一つの配列要素とみなして得られる４行４列の位置配列を表わす。

【０１２３】例として、図２５と同様に、Ｎ＝１２，Ｐ＝４とすると、Ｌ＝Ｎ／Ｐ＝３となり、式（１）は次の式となる。

(i,j)→((i-1)mod3+1, (j-1)mod3+1,[(i-1)/3]+1, [(J-1)/3]+1)　　（２）

ここで、データ配列２５００の左上角にある配列要素（１，１）は、式（２）より、（１，１，１，１）に移される。これは、位置配列の配列要素が（１，１）、即ち、データ配列２５００の左上角の太線で区切られた３行３列の部分であり、その３行３列の部分からなる２次元配列中の配列要素が（１，１）であることを示しており、元の（１，１）と同じアドレスを指していることがわかる。

【０１２４】また、データ配列２５００中のアドレス番号が８３の部分は、元の配列における配列要素が（５，７）であり、これは式（２）により、（２，１，２，３）に移される。これは、位置配列の配列要素が（２，３）、即ち、２５０２を示しており、その位置配列要素が指す、３行３列の部分からなる２次元配列中の配列要素が（２，１）であることを示しており、アドレス番号が８３の部分を指していることがわかる。

【０１２５】２５０１で示される、アドレス順で、１０番目から１８番目，４６番目から５４番目，８２番目から９０番目，１１８番目から１２６番目の、3行１２列の部分配列は、配列ａを（ｂｌｏｃｋ，＊）というデータ分散方法で分割して、本発明の並列化方法により並列化した時、Ｐ１がその配列要素の計算を担当するデータである。即ち、２５０２の配列要素の値の更新は、原則的にＰ１が行なう。２５１０で示される１２行１２列の配列は、配列ａを（＊，ｂｌｏｃｋ）というデータ分散方法で分割して、本発明の並列化方法により並列化した時、Ｐ２に割り付けられるデータである。この配列の大きさはデータ分散前の配列の大きさと同じである。

【０１２６】２５１０中の一部の数字も同様に、Ｐ２に割り付けられたデータ２５１０の配列要素がアドレス順に並ぶ様子を示している。２５１１で示される、アドレス順で、７３番目から１０８番目の、１２行３列の部分配列は、配列ａを（＊，ｂｌｏｃｋ）というデータ分散方法で分割して、本発明の並列化方法により並列化した時、Ｐ２がその配列要素の計算を担当するデータである。即ち、２５１１の配列要素の値の更新は、原則的にＰ２が行なう。

【０１２７】２５０１中の網掛け部分２５０２は、（ｂｌｏｃｋ，＊）から（＊，ｂｌｏｃｋ）へのデータ再分散により、Ｐ１からＰ２へ送信されるデータを表わす。図中に示されたアドレス順により、このデータは８２番目から９０番目までの配列要素からなる連続データであることがわかる。一方、２５１１中の網掛け部分２５１２は、（ｂｌｏｃｋ，＊）から（＊，ｂｌｏｃｋ）へのデータ再分散により、Ｐ２がＰ１から受信するデータを表わす。図中に示されたアドレス順により、このデータはやはり、８２番目から９０番目までの配列要素からなる連続データであることがわかる。

【０１２８】従って、送信側データも受信側データも連続となり、データ通信に際してはバッファを介する必要がないことがわかる。これは、配列からバッファへのコピーが不要であることを意味している。以上により、従来技術では、データ再分散に際し、データとバッファ間のコピーが必要であったが、本願により、それが不要になることがわかる。

【０１２９】尚、本実施例では、分散メモリマシン向けの並列化プログラムに関するデータ再分散の説明を行ったが、分散共有メモリマシンと呼ばれる、メモリは物理的に分散しているが、ハードウエアやＯＳなどにより、ユーザ側には共有メモリに見えるマシン向けの、並列化プログラムに関するデータ再分散に対しても本発明は適用可能である。

【０１３０】図２６は、本発明のコンパイラが対象とする並列計算機システムのハードウェア構成の一例を示したものである。同図において、２６０１はローカルメモリ、２６０２は論理プロセッサエレメント、２６０３はネットワーク、２６０４は入出力用論理プロセッサエレメント、２６０５は入出力用コンソールまたはワークステーションを表す。

【０１３１】本発明の並列化コンパイラ１００は、入出力用コンソールまたはワークステーション２６０５において実行され、並列ソースプログラムまたは並列オブジェクトプログラムに変換される。前者の並列ソースプログラムは、さらに、論理プロセッサエレメント２６０２向けのコンパイラにより並列オブジェクトプログラムに変換される。上記並列オブジェクトプログラムはリンカによりロードモジュールに変換され、入出力用論理プロセッサエレメント２６０４を通じて各論理プロセッサエレメント２６０２のローカルメモリ２６０１にロードされ、各論理プロセッサエレメント２６０２により実行さ

れる。実行時における各ロードモジュール間の通信はネットワーク２６０３を通じて行われる。本発明に係る並列化コンパイラは、上記並列計算機システムを有効利用してプログラムを高速化するものである。

【０１３２】（第２の実施例）次に、本発明の第２の実施例を図２７から図３３を用いて説明する。第２の実施例では共有メモリを持つ並列計算機を対象としている。図２７は、本発明による並列化コンパイラの構成を示したものである。２７００は並列化コンパイラである。並列化コンパイラ２７００中、ループ並列化解析部２７０１，データ分散形状決定部２７０２，プログラム並列化部２７０３以外は、図１のものと同じであるので説明を省略し、ループ並列化解析部２７０１，データ分散形状決定部２７０２，プログラム並列化部２７０３についてのみ説明する。

【０１３３】ループ並列化解析部２７０１は辞書１２０及び中間語１３０を入力し、ループの並列性を解析し、ループテーブル１５０を出力する。この処理の詳細は、ハンス・ジーマ、バーバラ・チャップマン著「スーパーコンパイラーズ フォー パラレル アンド ベトトル コンピューターズ」アディッソンーウエスリー、１９９１（Hans Zima and Barbara Chapman. "Supercompilers for Parallel and Vector Computers", Addison-Wesley, 1991）に説明されている。

【０１３４】データ分散形状決定部２７０２は辞書１２０，中間語１３０，及びループテーブル１５０を入力し、データ分散情報１４０を出力する。ここでは、まず、並列化された各ループに対して、そのループのループ制御変数が、配列のどの次元に出現するかを調べる。次に、ループ並列化によるループ繰り返しのプロセッサへの割り当て方法に対応して、配列要素がどのようなデータ分散方法によってプロセッサに割り付けられるかを調べる。これは、ループ中にある、代入文の左辺の配列添字に、並列化後の各プロセッサへ割り当てられるループ繰り返し範囲を代入することで得ることができる。

【０１３５】例えば、ループｉのループ繰り返しの１から１０，１１から２０，２１から３０，３１から４０回目が各々、プロセッサ０番，１番，２番，３番に割り当てられ、そのループ中で要素数４０の配列ａが、“ａ（ｉ）＝”の形で代入文の左辺に出現しているとすると、ａ（１）からａ（１０），ａ（１１）からａ（２０），ａ（２１）からａ（３０），ａ（３１）からａ（４０）までが各々、プロセッサ０番，１番，２番，３番によって、値が更新される。即ち、そのループの並列化方法は配列がブロック分散でプロセッサに割り付けられた時と同じになる。後は、上記の方法で得られた分散情報テーブル１４０を用いて、上述した第１の実施例と同じ方法を適用することで並列化プログラム２７２０が得られる。

【０１３６】次に、具体例を用いて説明する。図２８は、ソースプログラム２７１０を示す図である。これは、図８のソースプログラムから、プロセッサ指示文８０１、データ分散指示文８０２、データ再分散指示文８０４及び８１０を除いたものである。プロセッサ数はソースプログラム２７１０では指定されてないが、コンパイルオプションで４と指定されている、とする。

【０１３７】ループ並列化解析部２７０１は、ソースプログラム２７１０を解析し、第１のループネストである文２８０２から文２８０６に対してはループｉを並列化し、第２のループネストである文２８０７から文２８１１に対しては、ループｊを並列化する、と決定する。

【０１３８】データ分散形状決定部２７０２では、ループ並列化解析部２７０１が出力したループテーブル１５０，辞書１２０，及び中間語１３０を入力して以下の処理を行なう。まず、プロセッサ数は上記コンパイルオプションで４と指定されているので、第１のループネスト中のループｉでは、ループ繰り返しの１から３，４から６，７から９，１０から１２回目が各々、プロセッサ０番，１番，２番，３番に割り当てられており、この割り当て方法はデータ分散におけるブロック分散と同じである。

【０１３９】また、ループｉ中で１次元目の要素数１２の配列ａが、“ａ（ｉ，ｊ）＝”の形で代入文の左辺に出現しているので、ａ（１，ｊ）からａ（３，ｊ），ａ（４，ｊ）からａ（６，ｊ），ａ（７，ｊ）からａ（９，ｊ），ａ（１０，ｊ）からａ（１２，ｊ）までが各々、プロセッサ０番，１番，２番，３番によって、値が更新される。即ち、配列ａの１次元目はブロック分散される。同様にして、第２のループネスト中のループｊでは、ループ繰り返しがブロック分散され、ループｊ中で２次元目の要素数１２の配列ａが、“ａ（ｉ，ｊ）＝”の形で代入文の左辺に出現しているので、配列ａの２次元目はブロック分散される。よって、データ分散情報１４０には、第１のループネストでは、配列ａの１次元目がブロック分散され、第２のループネストでは配列ａの２次元目がブロック分散される、という情報が設定される。この結果、図９のデータ分散情報が得られる。形状収集部１０３からプログラム変換部１０７までの処理は前述した第１の実施例と同じである。

【０１４０】プログラム並列化部２７０３での処理も、ほぼ、第１の実施例と同じであるが、通信生成ステップ７０４（図７参照）におけるｒｅｍａｐやｓｅｎｄ，ｒｅｃｅｉｖｅ通信のかわりにバリア同期を引き起こすルーチンであるｂａｒｉｅｒを呼出し、ループを並列化するために、並列化部分の最初でスレッド生成ルーチンｆｏｒｋを、並列化部分の直後でスレッド消滅ルーチンｊｏｉｎを呼ぶ。

【０１４１】これらのｆｏｒｋ，ｊｏｉｎは、しばしば、並列化ループよりも外側に、しかも、複数の並列化ループに対して１組だけ呼出される。上記ｆｏｒｋとｊ

ｏｉｎによって囲まれる、実際の並列化ループよりも大きい範囲はＳＰＭＤ リージョンと呼ばれる。このＳＰＭＤリージョンの作成処理については、チャウ・ヴェン・ツェン著「コンパイラ オプティマイゼーション フォー エリミネイティングバリア シンクロナイゼーション」ピーピーオーピーピー９５、第１４４頁から第１５５頁、1995(Chau-Wen Tseng." Compiler Optimizations for Eliminationg Barrier Synchronization",PPOPP'95, pp.144-155, 1995)で説明されている。コード生成部１０９の処理も第１の実施例と同じである。

【０１４２】図２９は、上記の処理の結果得られた並列化プログラム２７２０を示す図である。文２９０３と文２９４０では、図２１の、第１の実施例における並列化プログラムにおける文２１０２及び文２１４０での再分散ルーチンのかわりに、バリア同期が呼出されている。文２９０１ではｆｏｒｋが、文２９８３ではｊｏｉｎが呼出されている。これらｆｏｒｋとｊｏｉｎで囲まれた部分がＳＰＭＤリージョンである。これら以外は図２１のプログラムと同じである。

【０１４３】図３０は従来技術による並列化プログラムを示す図である。文３００３と文３０２０では、やはり、図２３の、第１の実施例における並列化プログラムにおける文２３０２及び文２３２０での再分散ルーチンのかわりに、バリア同期が呼出されている。また、文３００１ではｆｏｒｋが、文３０４２ではｊｏｉｎが呼出されている。これら以外に、図２３と異なる点は以下の２つである。

【０１４４】図２３では、配列データが分散メモリに分散されていたため、一つのプロセッサにはそのうちの一部のみが割り付けられていた。そのため、図２３における配列ａの宣言はａ（３，１２）となっていた。しかし、図３０は共有メモリ型並列計算機に対するプログラムなので、配列ａは分散されず、宣言はａ（１２，１２）である。

【０１４５】また、上記と同じ理由により、図２３では、各プロセッサは、分散された配列の要素を参照するように、文２３１１及び文２３３０で示されるように、ループの繰り返し範囲も縮小されていた。ところが、図３０の文３０１１及び文３０３０では、各プロセッサは分散されない配列の一部分のみ参照するように、プロセッサ番号が格納された変数ｍｙｐｅを用いて、ちょうど自プロセッサが分担する配列要素のみ参照するようなループ範囲を生成している。

【０１４６】図３１は、従来の並列化によるプロセッサへの計算分担部分を、第１のループネストにおける論理プロセッサ１（以下、Ｐ１と略す）の計算分担部分と、第２のループネストにおける論理プロセッサ２（以下、Ｐ２と略す）の計算分担部分に注目して表示したものである。３１００は１２行１２列の配列ａ全体を示す。３１００中の数字は、配列要素がアドレス順に並ぶ様子を示している。左上角から始まる１２行３列の部分に、最初の３６要素（１～３６）のアドレス順が示されている。これにより、１列中の部分は連続アドレスであることがわかる。数字が表示されてない他の部分についても同様である。この本発明による配列要素のアドレスの変更は、図２５の場合と同じである。

【０１４７】４つの角のアドレスが、４，６，１３６，１３８番目になる矩形部分３１０１は、第１のループネストにおける、配列ａを（ｂｌｏｃｋ，＊）というデータ分散方法で分散することに対応した、Ｐ１の計算分担部分である。また、４つの角のアドレスが、３，４，１３５，１３６番目になる網掛け部分３１０２はプロセッサがキャッシュを持ち、そのキャッシュライン長が配列２要素分であり、配列ａの最初の要素がキャッシュライン先頭にある場合に、Ｐ１とＰ０が共有するキャッシュラインを示したものである。この条件の下では、キャッシュラインは、アドレスが奇数番目の配列要素から始まり、それより一つアドレスが大きい配列要素で終わる。したがって、２つのプロセッサは、１２個のキャッシュラインを共有し、フォールスシェアリングもその共有キャッシュラインの数だけ発生しやすい。

【０１４８】４つの角のアドレスが、７３，８４，９７，１０８番目になる矩形部分３１１１は、第２のループネストにおける、配列ａを（＊，ｂｌｏｃｋ）というデータ分散方法で分散することに対応した、Ｐ２の計算分担部分である。この矩形部分３１１１は、奇数番目のアドレスで始まり、偶数番目のアドレスで終わる単一の連続領域なので、他プロセッサとキャッシュを共有することはない。よって、フォールスシェアリングは第１のループネストにおいてのみ発生する。

【０１４９】図３２は、本発明の並列化によるプロセッサへの計算分担部分を、第１のループネストにおけるＰ１の計算分担部分と、第２のループネストにおけるＰ２の計算分担部分に注目して表示したものである。３２００は１２行１２列の配列ａ全体を示す。３２００中の数字は、配列要素がアドレス順に並ぶ様子を示している。左上角から始まる１２行３列の部分に、最初の３６要素（１～３６）のアドレス順が示されている。これにより、太線で囲まれた３行３列中の部分は連続アドレスであることがわかる。数字が表示されてない他の部分についても同様である。

【０１５０】４つの角のアドレスが、１０，１２，１２４，１２６番目になる矩形部分３２０１は、第１のループネストにおける、配列ａを（ｂｌｏｃｋ，＊）というデータ分散方法で分散することに対応した、Ｐ１の計算分担部分である。アドレスが９と１０，４５と４６，８１と８２，１１７と１１８の網掛け部分は、Ｐ１とＰ０が共有する４つのキャッシュラインを示したものである。図３１に比べて共有キャッシュラインの数が３分の１になっているので、フォールスシェアリングはそれだ

け、発生しにくい。

【０１５１】４つの角のアドレスが、７３，７９，１０２，１０８番目になる矩形部分３２１１は、第２のループネストにおける、配列ａを（＊，ｂｌｏｃｋ）というデータ分散方法で分散することに対応した、Ｐ２の計算分担部分である。この矩形部分３２１１は、奇数番目のアドレスで始まり、偶数番目のアドレスで終わる、単一の連続領域なので、他プロセッサとキャッシュを共有することはない。よって、フォールスシェアリングは第１のループネストにおいてのみ、発生する。

【０１５２】今の例では、共有キャッシュラインの数は３分の１になっていたが、一般に、従来技術では、図３１において配列ａの第２次元の寸法だけキャッシュラインを共有する可能性があり、本発明では、図３２において太線で囲まれた部分について１つだけ、即ち、プロセッサ数分だけキャッシュラインを共有する可能性があるので、第２次元の寸法がプロセッサ数を上回れば、後者の方がフォールスシェアリングは発生しにくい。

【０１５３】図３３は、本発明のコンパイラが対象とする並列計算機システムの構成の一例を示したものである。同図において、３３０１は共有メモリ、３３０２は論理プロセッサエレメント、３３０３は制御用ネットワーク、３３０４は入出力用論理プロセッサエレメント、３３０５は入出力用コンソールまたはワークステーションを表す。

【０１５４】本発明のコンパイラは、入出力用コンソールまたはワークステーション３３０５において実行され、並列ソースプログラムまたは並列オブジェクトプログラムに変換される。前者の並列ソースプログラムは、さらに、論理プロセッサエレメント３３０２向けのコンパイラにより並列オブジェクトプログラムに変換される。上記並列オブジェクトプログラムはリンカによりロードモジュールに変換され、入出力用論理プロセッサエレメント３３０４を通じて共有メモリ３３０１にロードされ、各論理プロセッサエレメント３３０２により実行される。論理プロセッサエレメント３３０２の起動、終了などの制御は制御用ネットワーク３３０３を通じて行われる。

【０１５５】

【発明の効果】本発明によれば、プロセッサｉからプロセッサｊへ送信される部分配列は、プロセッサｉ側でもプロセッサｊ側でも連続アドレスになるので、非連続データを１回の通信で送信するために使われるバッファが不要になり、バッファへのデータのコピー時間が削除されるので、その分、プログラムの実行時間が短縮される。

【０１５６】また、本発明によれば、各プロセッサはデータ分散前の配列全体を宣言して使用するので、データ再分散の際に、データ領域の確保・開放が不要であり、それにかかる実行時間が短縮される。また、本発明によれば、データ分散前の配列全体を宣言して使用しているため、再分散されるデータの内、自プロセッサから自プロセッサへ送信されるデータは、再分散前後でアドレスが全く変わらないため処理が不要であり、従来、自プロセッサ内の異なるアドレスへコピーしていたのにかかる時間が短縮される。

【０１５７】また、本発明によれば、プロセッサｉに割り付けられたデータは、送信先プロセッサによって、高々、プロセッサ総数分の非連続な領域に別れるだけなので、プロセッサｉに割り付けられたデータとプロセッサｊに割り付けられたデータのアドレスが隣合う回数は、全プロセッサ数以下となる。したがって、異なるプロセッサが同じキャッシュラインを共有する機会が少ないため、異なるプロセッサが同じキャッシュラインをアクセスすることにより発生するフォールスシェアリングの発生する機会が少なく、プログラムの実行時間が短縮される。

【図面の簡単な説明】

【図１】本発明の第１実施例による並列化コンパイラの構成図である。

【図２】配列細分化部の処理を説明するためのフローチャートである。

【図３】配列高次元化部の処理を説明するためのフローチャートである。

【図４】ループ変換部の処理を説明するためのフローチャートである。

【図５】ループピーリング解析処理を説明するためのフローチャートである。

【図６】添字変更部の処理を説明するためのフローチャートである。

【図７】プログラム並列化部の処理を説明するためのフローチャートである。

【図８】ソースプログラムの例を示す図である。

【図９】ソースプログラムに対するデータ分散情報を説明するための図である。

【図１０】ソースプログラムに対する辞書を説明するための図である。

【図１１】配列細分化による細分化情報を説明するための図である。

【図１２】配列高次元化による高次元化辞書を説明するための図である。

【図１３】ソースプログラムに対するループテーブルとループ分散範囲テーブルを説明するための図である。

【図１４】ソースプログラムに対する配列参照テーブルとピーリングテーブルを説明するための図である。

【図１５】ループ範囲細分化後のループ分散範囲テーブルを説明するための図である。

【図１６】ループｉのループ範囲細分化後のループテーブルを説明するための図である。

【図１７】ループｊのループ範囲細分化後のループテー

ブルを説明するための図である。
【図１８】ループピーリング後の中間語をプログラムイメージで表現した図である。
【図１９】添字変更後の中間語をプログラムイメージで表現した図である。
【図２０】データ分散情報変更後のデータ分散情報を説明するための図である。
【図２１】並列化後のプログラムを示した図である。
【図２２】データ分散前の配列によるデータ再分散の様子を示した図である。
【図２３】従来技術による並列化プログラムを示した図である。
【図２４】従来の並列化によるデータ再分散を説明するための図である。
【図２５】本発明の第１実施例の並列化によるデータ再分散を説明するための図である。
【図２６】本発明の第１実施例のコンパイラが対象とする並列計算機システムの構成の一例である。
【図２７】本発明の第２実施例による並列化コンパイラの構成図である。
【図２８】ソースプログラムの例を示す図である。
【図２９】並列化後のプログラムを示す図である。
【図３０】従来技術による並列化プログラムを示す図である。
【図３１】従来の並列化によるプロセッサへの計算分担部分を説明するための図である。
【図３２】本発明の第２実施例の並列化によるプロセッサへの計算分担部分を説明するための図である。
【図３３】本発明の第２実施例のコンパイラが対象とする並列計算機システムの構成の一例を示す図である。
【符号の説明】

＊１００：並列化コンパイラ
１０１：構文解析部
１０２：データ分散指示文解析部
１０３：形状収集部
１０４：形状種別数判定部
１０５：配列細分化部
１０６：配列高次元化部
１０７：プログラム変換部
１０７１：ループ変換部
１０７２：添字変更部
１０８：プログラム並列化部
１０９：コード生成部
１１０，２７１０：ソースプログラム
１２０：辞書
１３０：中間語
１４０：データ分散情報
１５０：ループテーブル
１６０，２７２０：並列化プログラム
２６０１：ローカルメモリ
２６０２，３３０２：論理プロセッサエレメント
２６０３：ネットワーク
２６０４，３３０４：入出力用論理プロセッサエレメント
２６０５，３３０５：入出力用コンソールまたはワークステーション
２７０１：ループ並列化解析部
２７０２：データ分散形状決定部
２７０３：プログラム並列化部
３３０１：共有メモリ
３３０３：制御用ネットワーク

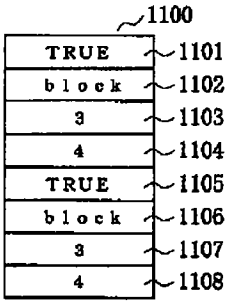
【図８】

ソースプログラム　110

```
800:      real a(12,12),b,c
801:!HPF$ PROCESSORS proc(4)
802:!HPF$ DISTRIBUTE a(block,*) ONTO proc
803:      do k = 1,10
804:!HPF$ REDISTRIBUTE a(block,*) ONTO proc
805:        do j = 2,12
806:        do i = 1,12
807:          a(i,j)=a(i,j)+a(i,j-1)*b
808:        enddo
809:        enddo
810:!HPF$ REDISTRIBUTE a(*,block) ONTO proc
811:        do j = 1,12
812:        do i = 2,12
813:          a(i,j)=a(i,j)+a(i-1,j)*c
814:        enddo
815:        enddo
816:        call update(b,c)
817:      enddo
```

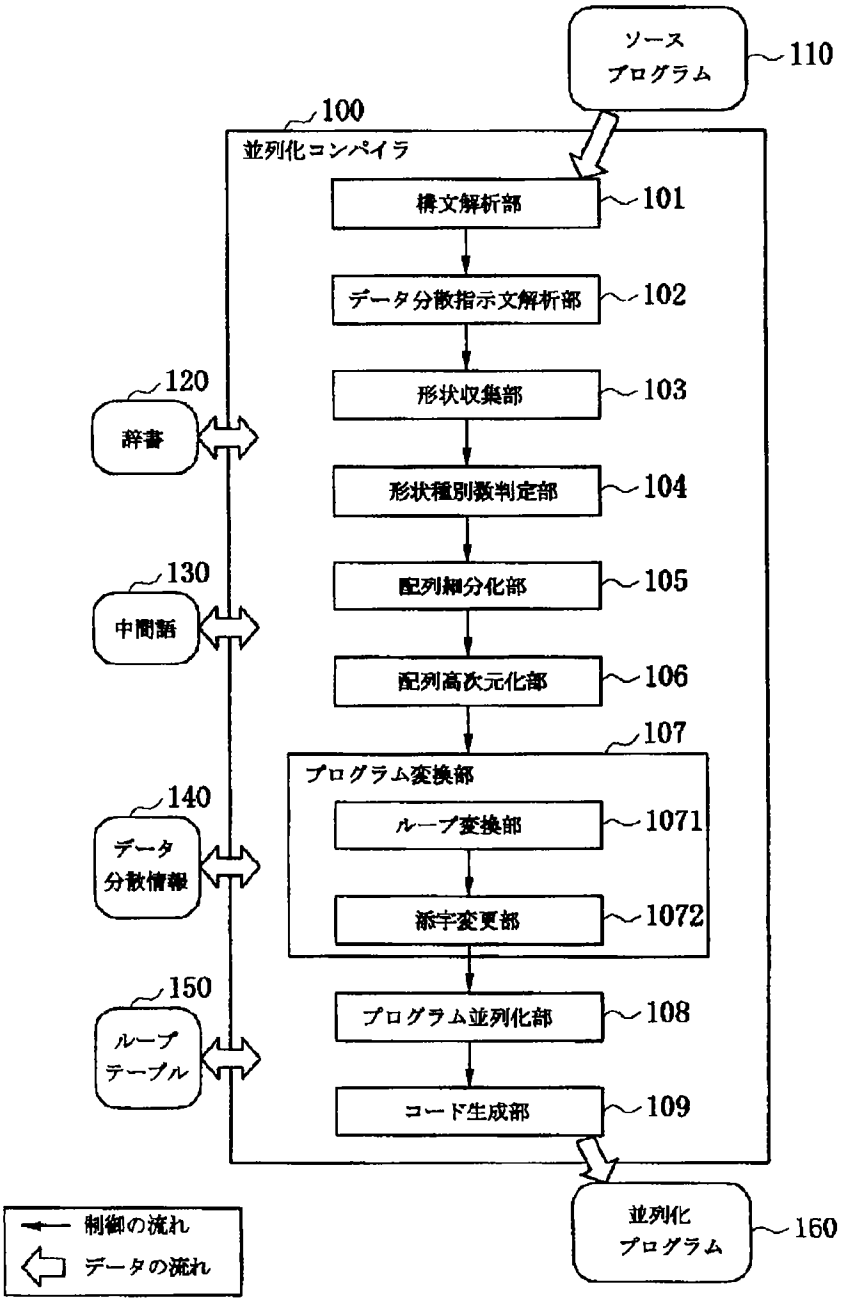
【図１１】

細分化情報　1100

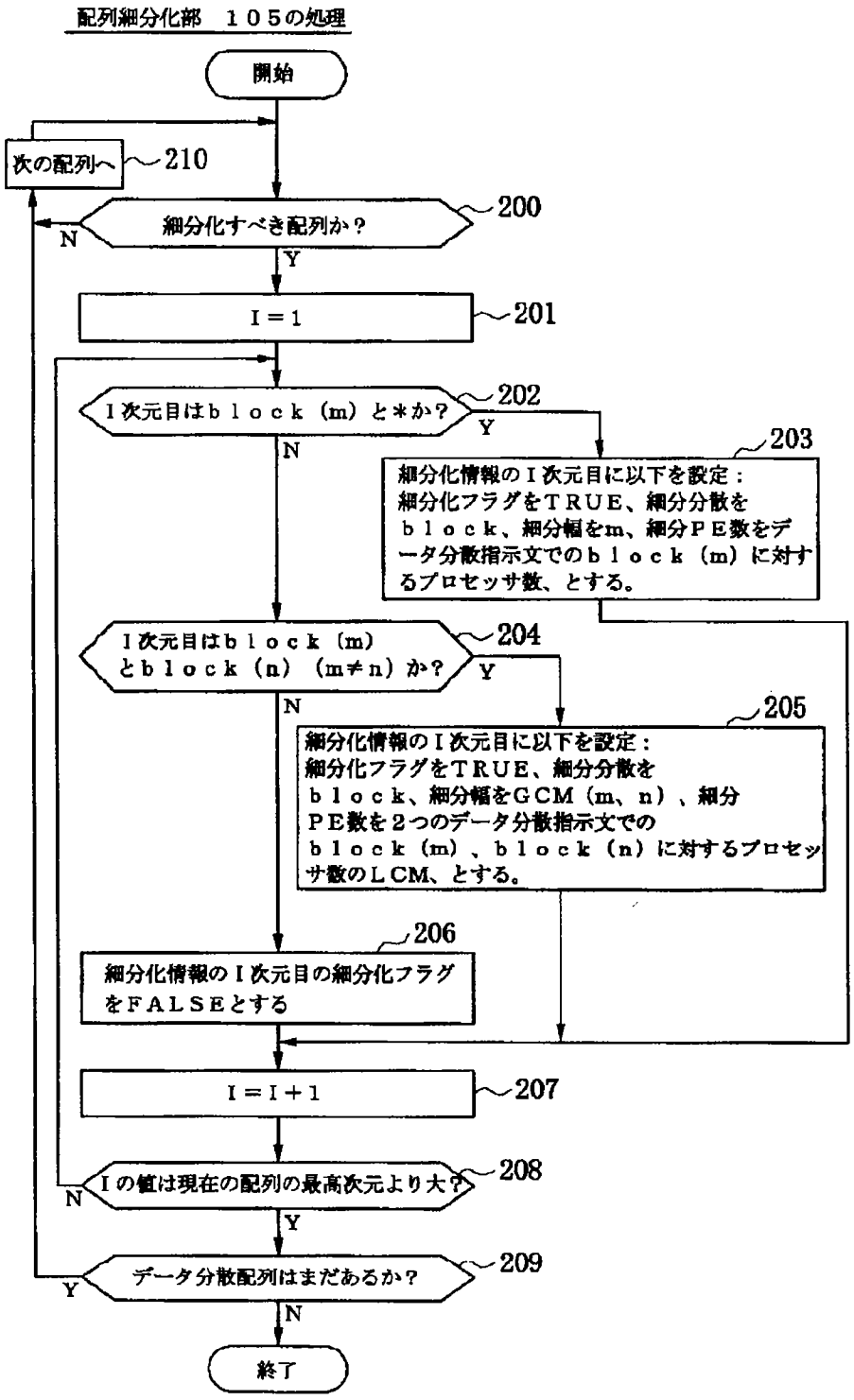


【図１】

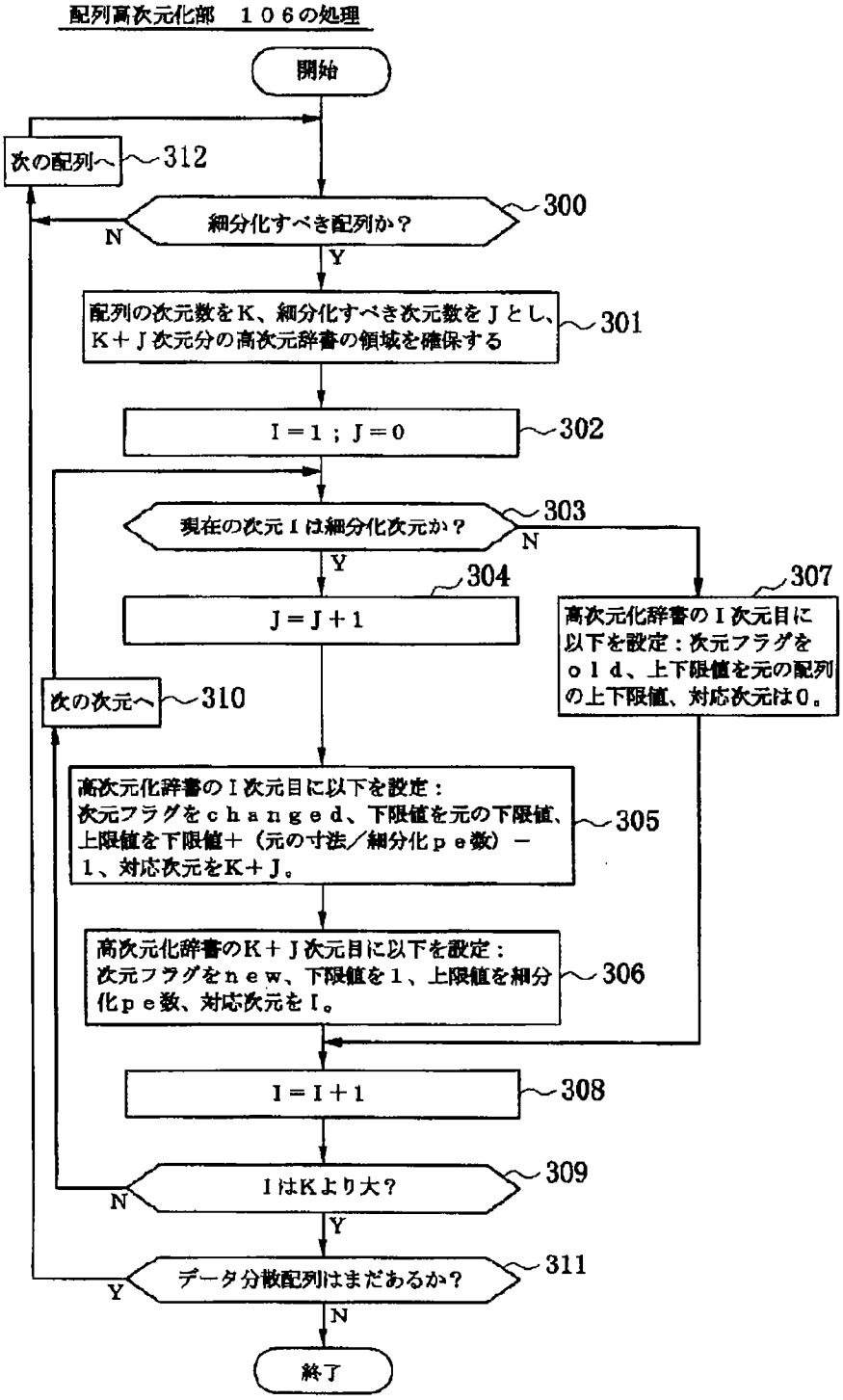
構成


ソース
プログラム
110
100
並列化コンパイラ
構文解析部
101
データ分散指示文解析部
102
形状収集部
103
形状種別数判定部
104
配列細分化部
105
配列高次元化部
106
107
プログラム変換部
ループ変換部
1071
添字変更部
1072
プログラム並列化部
108
コード生成部
109
120
辞書
130
中間語
140
データ
分散情報
150
ループ
テーブル
並列化
プログラム
160
制御の流れ
データの流れ

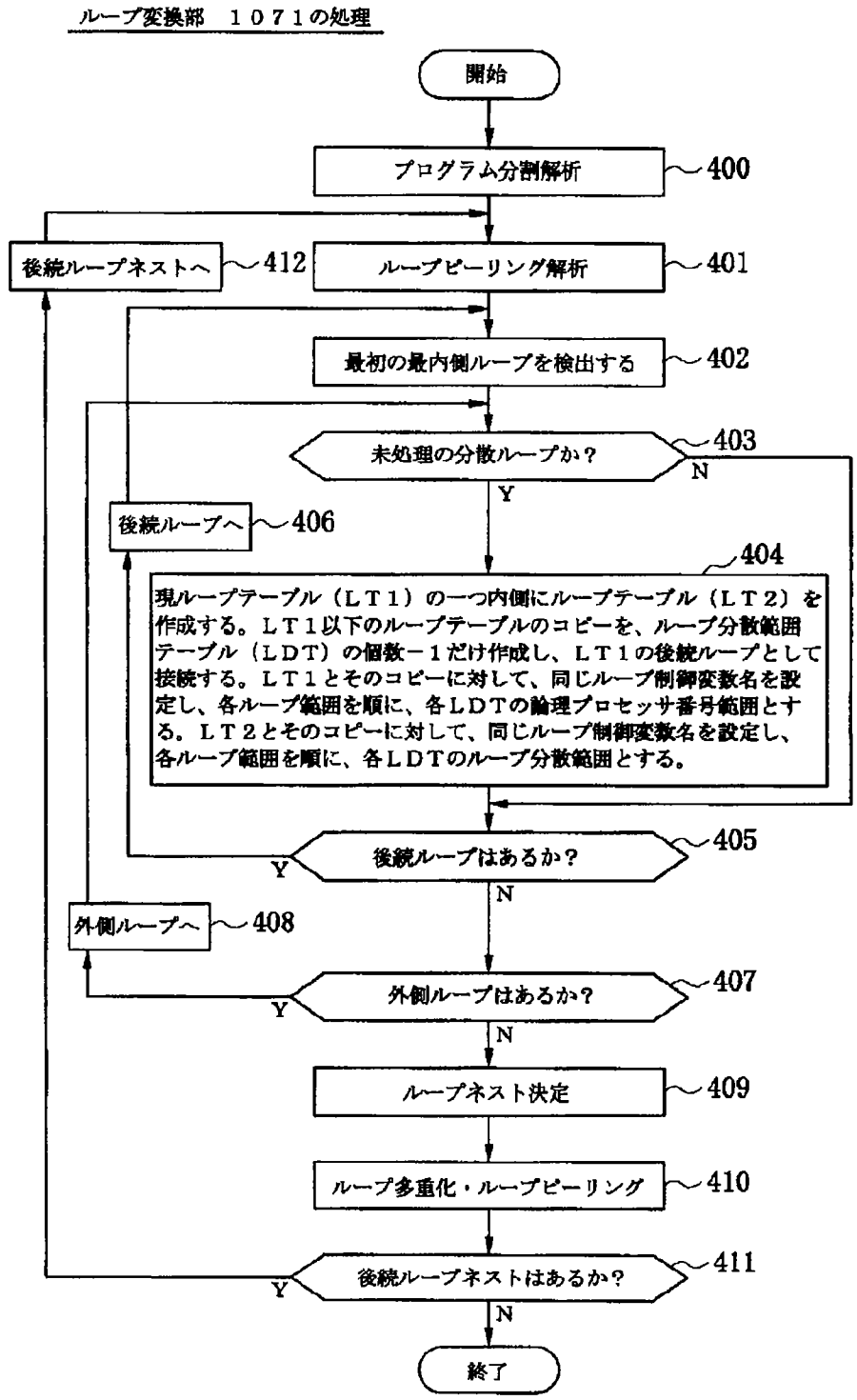
【図２】


配列細分化部　１０５の処理
開始
次の配列へ
210
細分化すべき配列か？
200
N
Y
Ｉ＝１
201
Ｉ次元目はｂｌｏｃｋ（ｍ）と＊か？
202
Y
N
203
細分化情報のＩ次元目に以下を設定：
細分化フラグをＴＲＵＥ、細分分散を
ｂｌｏｃｋ、細分幅をm、細分ＰＥ数をデータ分散指示文でのｂｌｏｃｋ（ｍ）に対するプロセッサ数、とする。
Ｉ次元目はｂｌｏｃｋ（ｍ）
とｂｌｏｃｋ（ｎ）（ｍ≠ｎ）か？
204
Y
N
205
細分化情報のＩ次元目に以下を設定：
細分化フラグをＴＲＵＥ、細分分散を
ｂｌｏｃｋ、細分幅をＧＣＭ（ｍ、ｎ）、細分
ＰＥ数を２つのデータ分散指示文での
ｂｌｏｃｋ（ｍ）、ｂｌｏｃｋ（ｎ）に対するプロセッサ数のＬＣＭ、とする。
206
細分化情報のＩ次元目の細分化フラグ
をＦＡＬＳＥとする
Ｉ＝Ｉ＋１
207
Ｉの値は現在の配列の最高次元より大？
208
N
Y
データ分散配列はまだあるか？
209
Y
N
終了

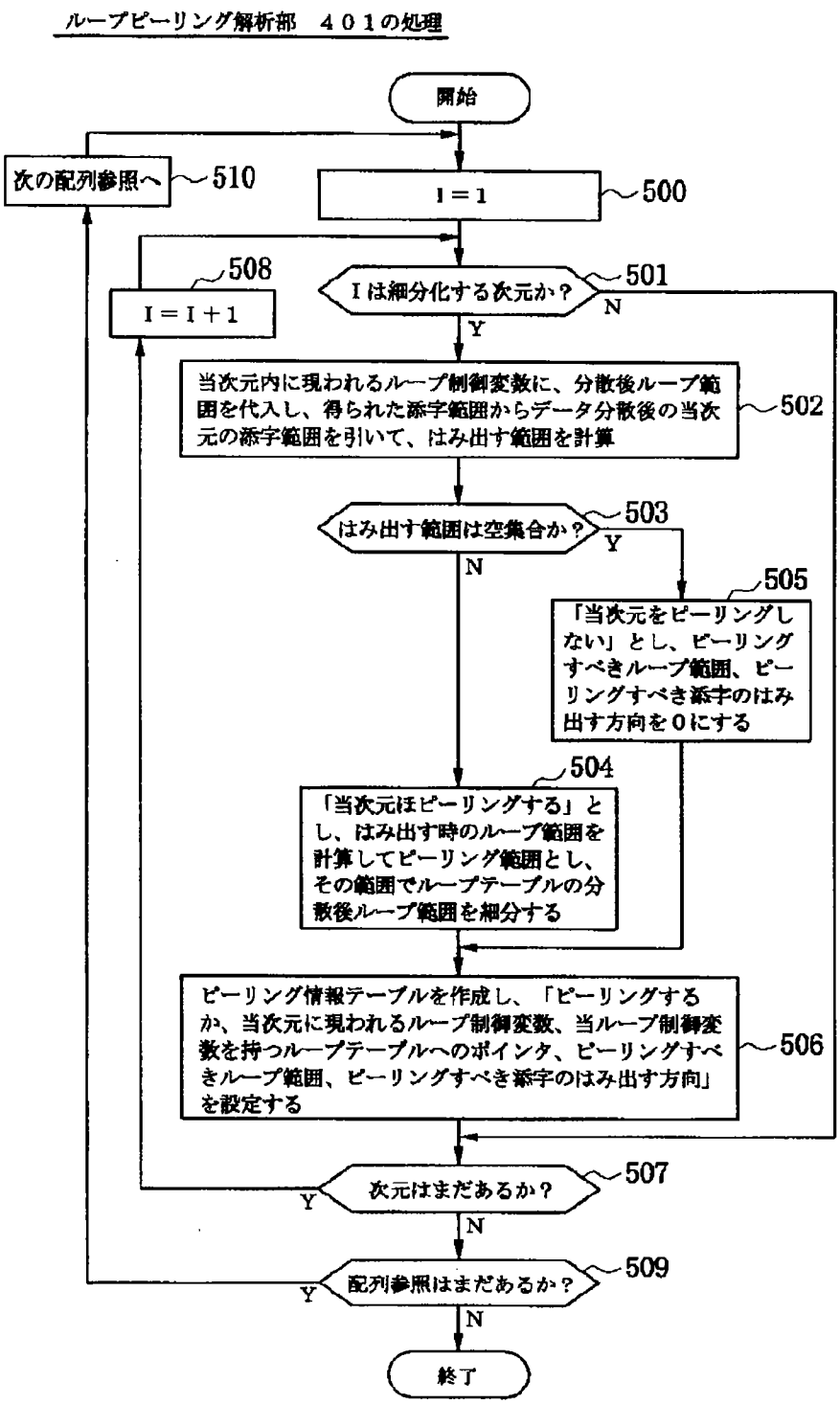
【図３】


配列高次元化部　１０６の処理
開始
次の配列へ
312
細分化すべき配列か？
300
N
Y
配列の次元数をＫ、細分化すべき次元数をＪとし、Ｋ＋Ｊ次元分の高次元辞書の領域を確保する
301
Ｉ＝１；Ｊ＝０
302
現在の次元Ｉは細分化次元か？
303
N
Y
Ｊ＝Ｊ＋１
304
高次元化辞書のＩ次元目に以下を設定：次元フラグをｏｌｄ、上下限値を元の配列の上下限値、対応次元は０。
307
次の次元へ
310
高次元化辞書のＩ次元目に以下を設定：次元フラグをｃｈａｎｇｅｄ、下限値を元の下限値、上限値を下限値＋（元の寸法／細分化ｐｅ数）－１、対応次元をＫ＋Ｊ。
305
高次元化辞書のＫ＋Ｊ次元目に以下を設定：次元フラグをｎｅｗ、下限値を１、上限値を細分化ｐｅ数、対応次元をＩ。
306
Ｉ＝Ｉ＋１
308
ＩはＫより大？
309
N
Y
データ分散配列はまだあるか？
311
Y
N
終了

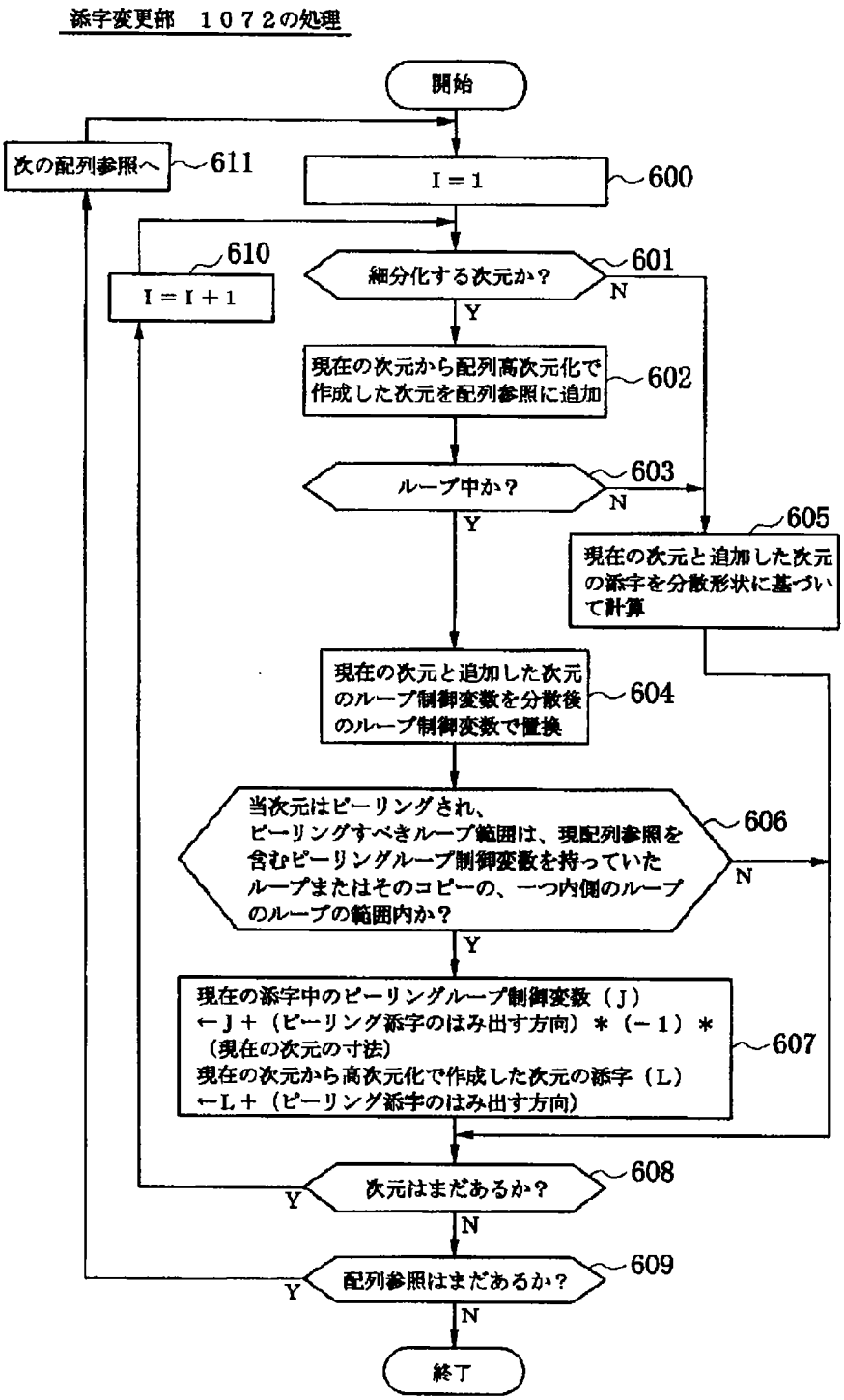
【図４】


ループ変換部　１０７１の処理
開始
プログラム分割解析
400
後続ループネストへ
412
ループピーリング解析
401
最初の最内側ループを検出する
402
未処理の分散ループか？
403
N
Y
後続ループへ
406
404
現ループテーブル（ＬＴ１）の一つ内側にループテーブル（ＬＴ２）を作成する。ＬＴ１以下のループテーブルのコピーを、ループ分散範囲テーブル（ＬＤＴ）の個数－１だけ作成し、ＬＴ１の後続ループとして接続する。ＬＴ１とそのコピーに対して、同じループ制御変数名を設定し、各ループ範囲を順に、各ＬＤＴの論理プロセッサ番号範囲とする。ＬＴ２とそのコピーに対して、同じループ制御変数名を設定し、各ループ範囲を順に、各ＬＤＴのループ分散範囲とする。
後続ループはあるか？
405
Y
N
外側ループへ
408
外側ループはあるか？
407
Y
N
ループネスト決定
409
ループ多重化・ループピーリング
410
後続ループネストはあるか？
411
Y
N
終了

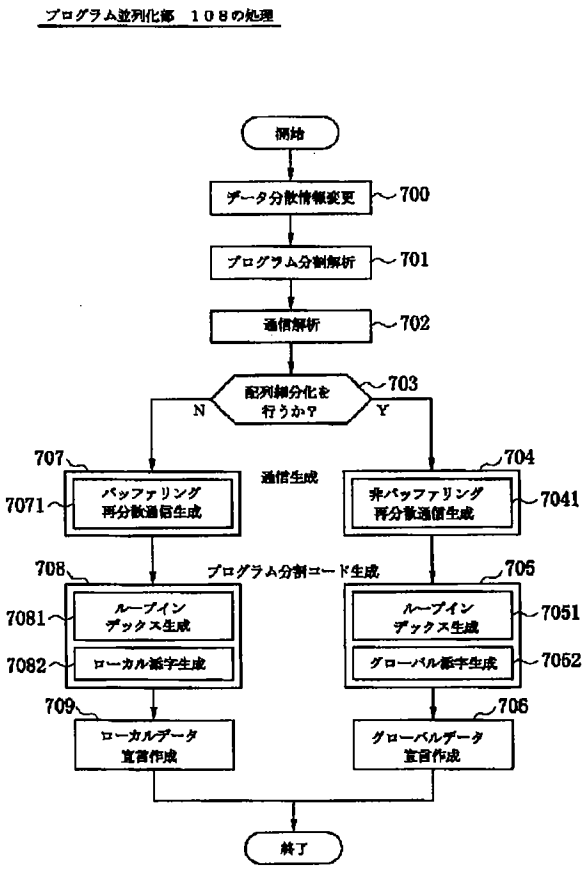
【図５】


ループピーリング解析部　４０１の処理
開始
I＝1
500
I は細分化する次元か？
501
N
Y
当次元内に現われるループ制御変数に、分散後ループ範囲を代入し、得られた添字範囲からデータ分散後の当次元の添字範囲を引いて、はみ出す範囲を計算
502
はみ出す範囲は空集合か？
503
Y
N
「当次元をピーリングしない」とし、ピーリングすべきループ範囲、ピーリングすべき添字のはみ出す方向を０にする
505
「当次元はピーリングする」とし、はみ出す時のループ範囲を計算してピーリング範囲とし、その範囲でループテーブルの分散後ループ範囲を細分する
504
ピーリング情報テーブルを作成し、「ピーリングするか、当次元に現われるループ制御変数、当ループ制御変数を持つループテーブルへのポインタ、ピーリングすべきループ範囲、ピーリングすべき添字のはみ出す方向」を設定する
506
次元はまだあるか？
507
Y
N
配列参照はまだあるか？
509
Y
N
終了
次の配列参照へ
510
I＝I＋1
508

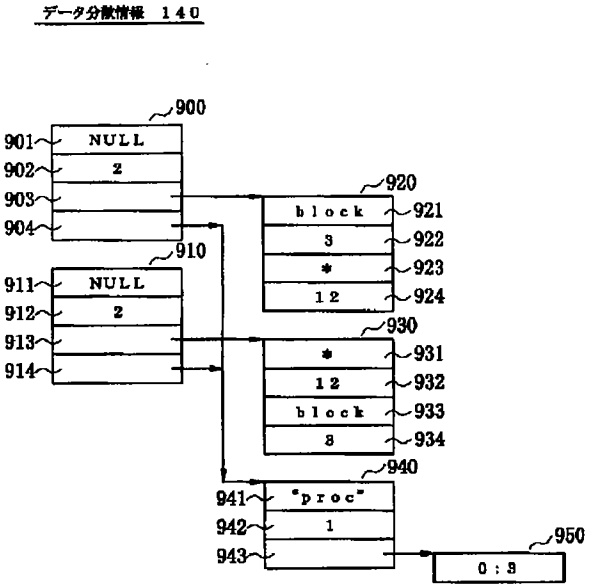
【図６】


添字変更部　１０７２の処理
開始
次の配列参照へ
611
I＝1
600
610
I＝I＋1
細分化する次元か？
601
N
Y
現在の次元から配列高次元化で作成した次元を配列参照に追加
602
ループ中か？
603
N
Y
605
現在の次元と追加した次元の添字を分散形状に基づいて計算
現在の次元と追加した次元のループ制御変数を分散後のループ制御変数で置換
604
当次元はピーリングされ、ピーリングすべきループ範囲は、現配列参照を含むピーリングループ制御変数を持っていたループまたはそのコピーの、一つ内側のループのループの範囲内か？
606
N
Y
現在の添字中のピーリングループ制御変数（Ｊ）←Ｊ＋（ピーリング添字のはみ出す方向）＊（－１）＊（現在の次元の寸法）
現在の次元から高次元化で作成した次元の添字（Ｌ）←Ｌ＋（ピーリング添字のはみ出す方向）
607
次元はまだあるか？
608
Y
N
配列参照はまだあるか？
609
Y
N
終了

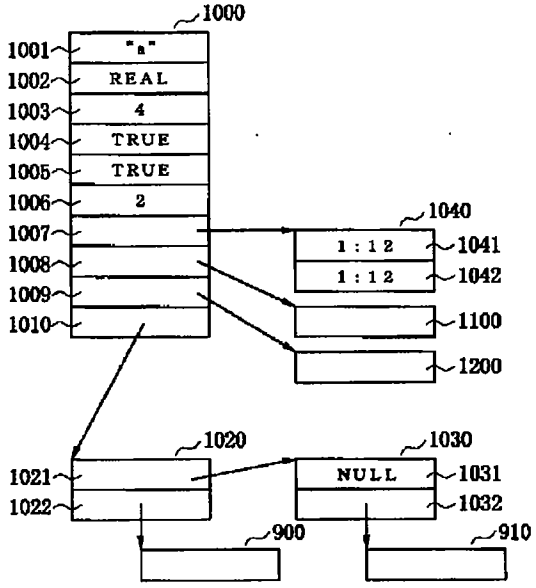
【図７】


プログラム並列化部　１０８の処理
開始
データ分散情報変更
700
プログラム分割解析
701
通信解析
702
配列細分化を行うか？
703
N
Y
通信生成
707
バッファリング再分散通信生成
7071
704
非バッファリング再分散通信生成
7041
プログラム分割コード生成
708
ループインデックス生成
7081
ローカル添字生成
7082
705
ループインデックス生成
7051
グローバル添字生成
7052
709
ローカルデータ宣言作成
706
グローバルデータ宣言作成
終了

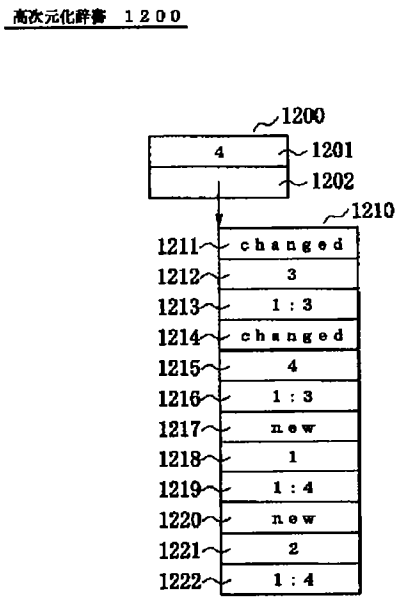
【図９】


データ分散情報　１４０
900
901
NULL
902
2
903
904
920
block
921
3
922
*
923
12
924
910
911
NULL
912
2
913
914
930
*
931
12
932
block
933
3
934
940
941
"proc"
942
1
943
950
0：3

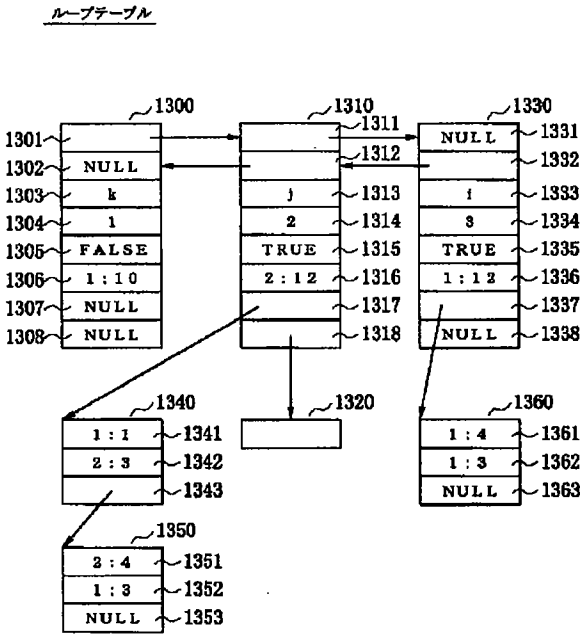
【図１０】


辞書　１２０
1000
1001
"a"
1002
REAL
1003
4
1004
TRUE
1005
TRUE
1006
2
1007
1008
1009
1010
1040
1：12
1041
1：12
1042
1100
1200
1020
1021
1022
1030
NULL
1031
1032
900
910

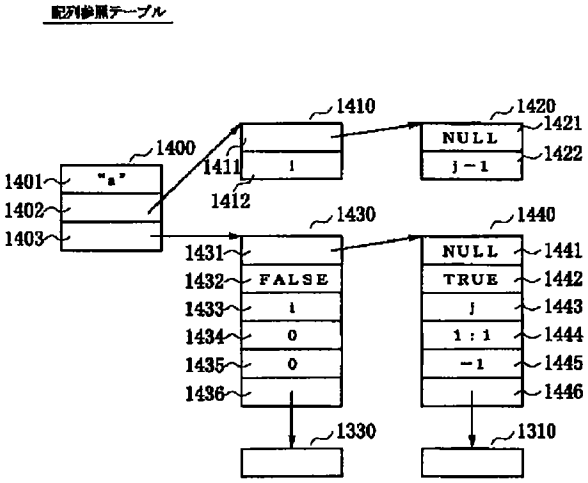
【図１２】


高次元化辞書　１２００
1200
4
1201
1202
1210
1211
changed
1212
3
1213
1：3
1214
changed
1215
4
1216
1：3
1217
new
1218
1
1219
1：4
1220
new
1221
2
1222
1：4

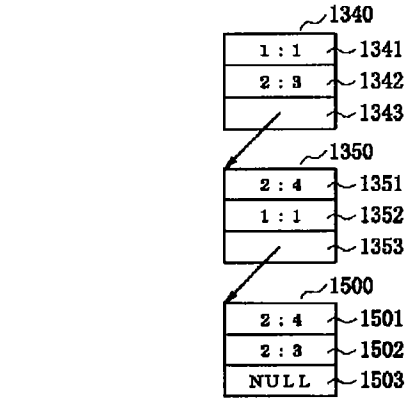
【図１３】



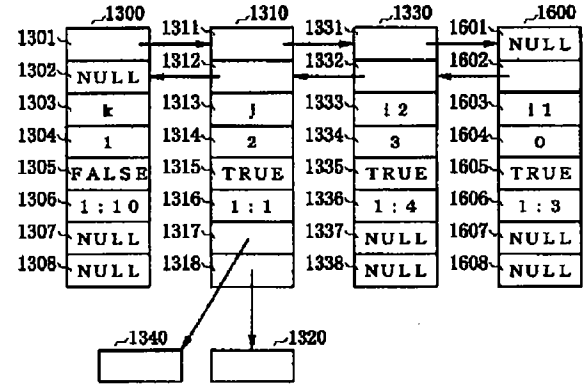
【図１４】



【図１５】

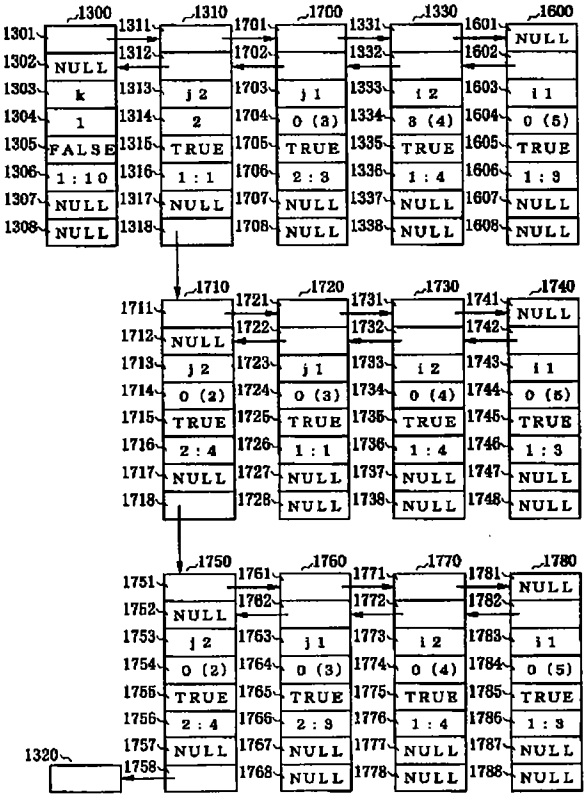


【図１６】



【図１７】

ループjのループ範囲細分化後のループテーブル



【図１８】

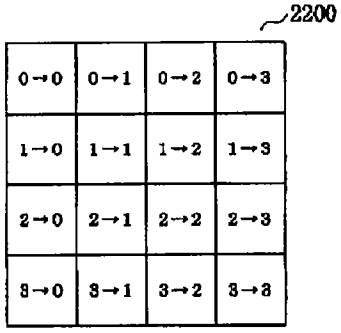
ループピーリング後のプログラム

```
1800:do k = 1,10
1810:   do j2 = 1,1
1811:   do j1 = 2,3
1812:   do i2 = 1,4
1813:   do i1 = 1,3
1814:     a(i,j)=a(i,j)+ a(i,j-1)*b
1815:   enddo;enddo;enddo;enddo
1820:   do j2 = 2,4
1821:   do j1 = 1,1
1822:   do i2 = 1,4
1823:   do i1 = 1,3
1824:     a(i,j)=a(i,j)+ a(i,j-1)*b
1825:   enddo;enddo;enddo;enddo
1830:   do j2 = 2,4
1831:   do j1 = 2,3
1832:   do i2 = 1,4
1833:   do i1 = 1,3
1834:     a(i,j)=a(i,j)+ a(i,j-1)*b
1835:   enddo;enddo;enddo;enddo

1840:   do j2 = 1,4
1841:   do j1 = 1,3
1850:   do i2 = 1,1
1851:   do i1 = 2,3
1852:     a(i,j)=a(i,j)+ a(i-1,j)*c
1853:   enddo;enddo
1860:   do i2 = 2,4
1861:   do i1 = 1,1
1862:     a(i,j)=a(i,j)+ a(i-1,j)*c
1863:   enddo;enddo
1870:   do i2 = 2,4
1871:   do i1 = 2,3
1872:     a(i,j)=a(i,j)+ a(i-1,j)*c
1873:   enddo;enddo
1880:   enddo;enddo
1881:   call update(b,c)
1882:enddo
```

【図２２】

データ分散前の配列によるデータ再分散の様子



【図２３】

従来技術による並列化プログラム

```
2300:real a(3,12),b,c
2301:do k = 1,10
2302:   call remap(a,E1,E2)
2310:   do j = 2,12
2311:   do i = 1,3
2313:     a(i,j)=a(i,j)+ a(i,j-1)*b
2314:   enddo;enddo
2320:   call remap(a,E2,E1)
2330:   do j = 1,3
2331:   do i = 2,12
2332:     a(i,j)=a(i,j)+ a(i-1,j)*c
2333:   enddo;enddo
2340:   call update(b,c)
2341:enddo
```

【図１９】

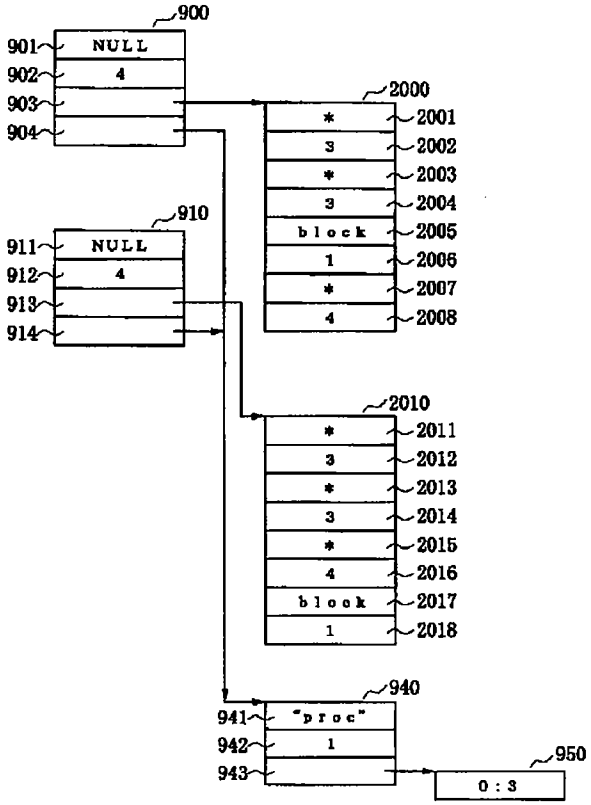
添字変更後のプログラム

```
1900:real a(3,3,4,4)
1901:do k = 1,10
1910:   do j2 = 1,1
1911:   do j1 = 2,3
1912:   do i2 = 1,4
1913:   do i1 = 1,3
1914:     a(i1,j1,i2,j2)=a(i1,j1,i2,j2)+ &
1915:         a(i1,j1-1,i2,j2)*b
1916:   enddo;enddo;enddo;enddo
1920:   do j2 = 2,4
1921:   do j1 = 1,1
1922:   do i2 = 1,4
1923:   do i1 = 1,3
1924:     a(i1,j1,i2,j2)=a(i1,j1,i2,j2)+ &
1925:         a(i1,j1+2,i2,j2-1)*b
1926:   enddo;enddo;enddo;enddo
1930:   do j2 = 2,4
1931:   do j1 = 2,3
1932:   do i2 = 1,4
1933:   do i1 = 1,3
1934:     a(i1,j1,i2,j2)=a(i1,j1,i2,j2)+ &
1935:         a(i1,j1-1,i2,j2)*b
1936:   enddo;enddo;enddo;enddo

1940:   do j2 = 1,4
1941:   do j1 = 1,3
1950:   do i2 = 1,1
1951:   do i1 = 2,3
1952:     a(i1,j1,i2,j2)=a(i1,j1,i2,j2)+ &
1953:         a(i1-1,j1,i2,j2)*c
1954:   enddo;enddo
1960:   do i2 = 2,4
1961:   do i1 = 1,1
1962:     a(i1,j1,i2,j2)=a(i1,j1,i2,j2)+ &
1963:         a(i1+2,j1,i2-1,j2)*c
1964:   enddo;enddo
1970:   do i2 = 2,4
1971:   do i1 = 2,3
1972:     a(i1,j1,i2,j2)=a(i1,j1,i2,j2)+ &
1973:         a(i1-1,j1,i2,j2)*c
1974:   enddo;enddo
1980:   enddo;enddo
1981:   call update(b,c)
1982:enddo
```

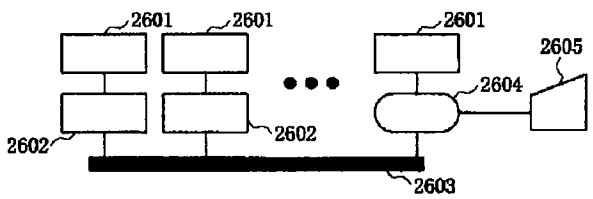
【図２０】

変更後のデータ分散情報



【図２６】

並列計算機システムの一例



【図２８】

入力プログラム　2710

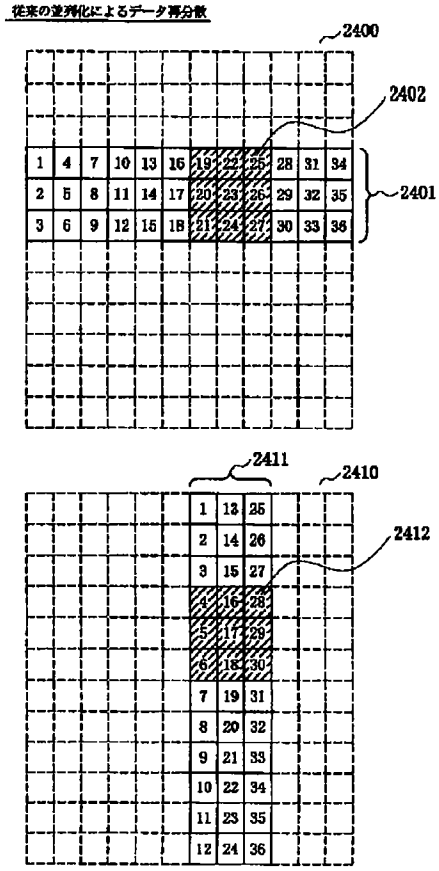
```
2800:       real a(12,12),b,c
2801:       do k = 1,10
2802:         do j = 2,12
2803:         do i = 1,12
2804:          a(i,j)=a(i,j)+a(i,j-1)*b
2805:         enddo
2806:         enddo
2807:         do j = 1,12
2808:         do i = 2,12
2809:           a(i,j)=a(i,j)+a(i-1,j)*c
2810:         enddo
2811:         enddo
2812:         call update(b,c)
2813:       enddo
```

【図２１】

並列化後のプログラム　160

```
2100:real a(3,3,4,4),b,c
2101:do k = 1,10
2102:   call non_buf_remap(a,D1,D2)
2110:   do j2 = 1,1
2111:   do j1 = 2,3
2112:   do i1 = 1,3
2113:     a(i1,j1,mype,j2)=a(i1,j1,mype,j2)+ &
2114:         a(i1,j1-1,mype,j2)*b
2115:   enddo;enddo;enddo
2120:   do j2 = 2,4
2121:   do j1 = 1,1
2122:   do i1 = 1,3
2123:     a(i1,j1,mype,j2)=a(i1,j1,mype,j2)+ &
2124:         a(i1,j1+2,mype,j2-1)*b
2125:   enddo;enddo;enddo
2130:   do j2 = 2,4
2131:   do j1 = 2,3
2132:   do i1 = 1,3
2133:     a(i1,j1,mype,j2)=a(i1,j1,mype,j2)+ &
2134:         a(i1,j1-1,mype,j2)*b
2135:   enddo;enddo;enddo
2140:   call non_buf_remap(a,D2,D1)
2141:   do j1 = 1,3
2150:   do i2 = 1,1
2151:   do i1 = 2,3
2152:     a(i1,j1,i2,mype)=a(i1,j1,i2,mype)+ &
2153:         a(i1-1,j1,i2,mype)*c
2154:   enddo;enddo
2160:   do i2 = 2,4
2161:   do i1 = 1,1
2162:     a(i1,j1,i2,mype)=a(i1,j1,i2,mype)+ &
2163:         a(i1+2,j1,i2-1,mype)*c
2164:   enddo;enddo
2170:   do i2 = 2,4
2171:   do i1 = 2,3
2172:     a(i1,j1,i2,mype)=a(i1,j1,i2,mype)+ &
2173:         a(i1-1,j1,i2,mype)*c
2174:   enddo;enddo
2180:   enddo
2181:   call update(b,c)
2182:enddo
```

【図２４】

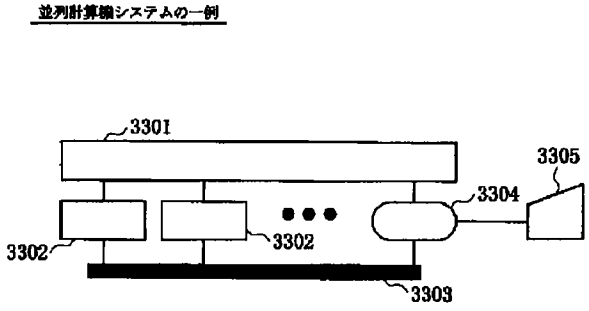


【図３０】

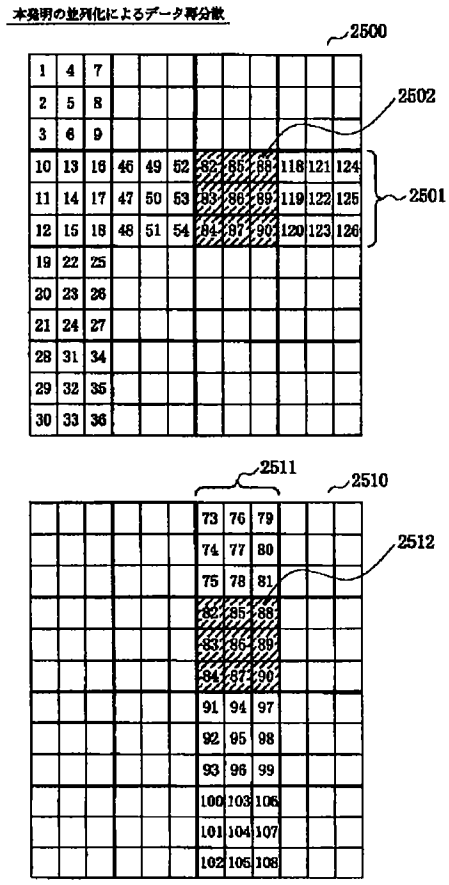
従来技術による並列化プログラム

```
3000:real a(12,12),b,c
3001:call fork
3002:do k = 1,10
3003:   call barrier
3010:   do j = 2,12
3011:   do i = 3*mype+1,3*(mype+1)
3012:     a(i,j)=a(i,j)+ a(i,j-1)*b
3013:   enddo;enddo
3020:   call barrier
3030:   do j = 3*mype+1,3*(mype+1)
3031:   do i = 2,12
3032:     a(i,j)=a(i,j)+ a(i-1,j)*c
3033:   enddo;enddo
3040:   call update(b,c)
3041:enddo
3042:call join
```

【図３３】



【図２５】

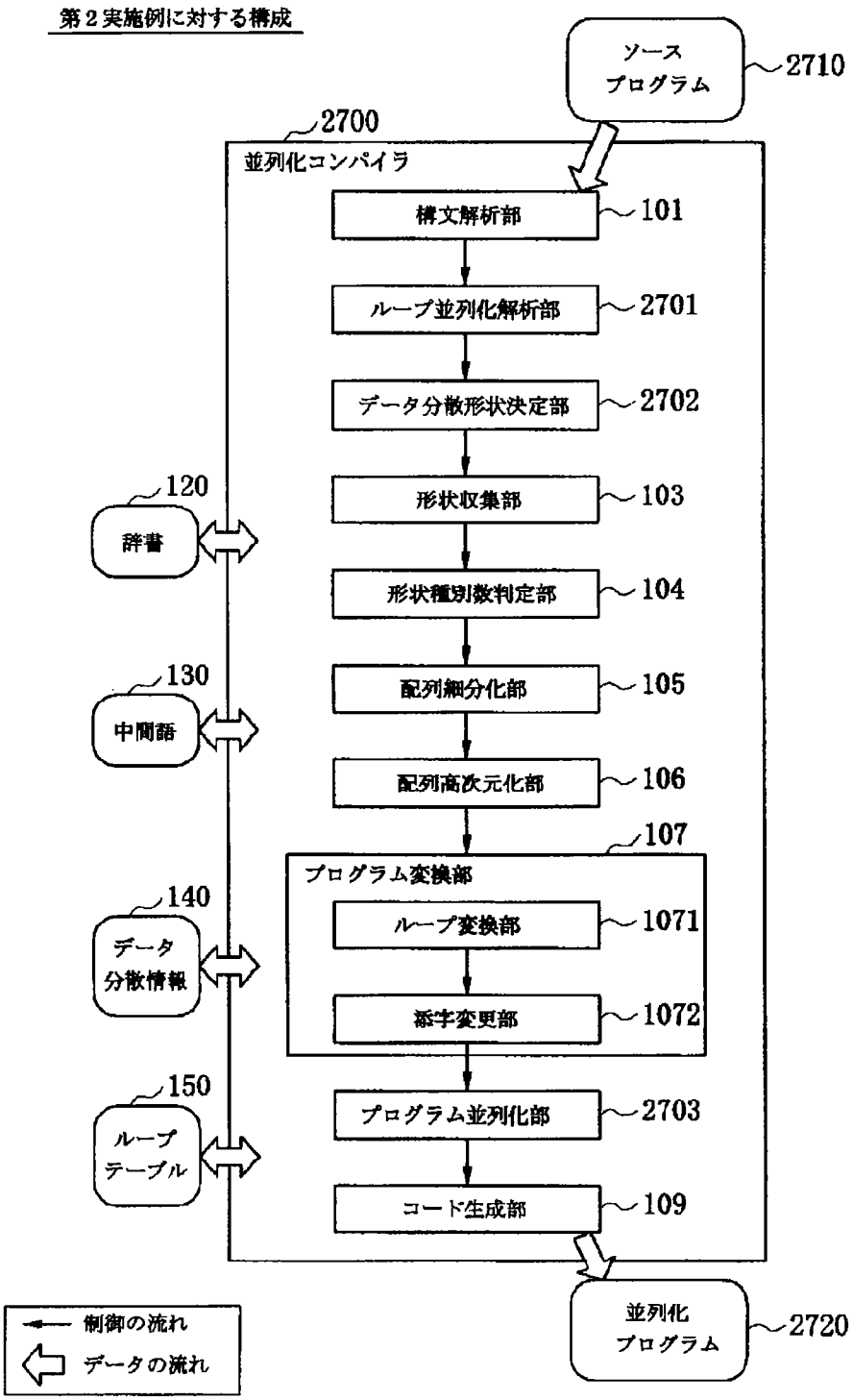


【図２９】

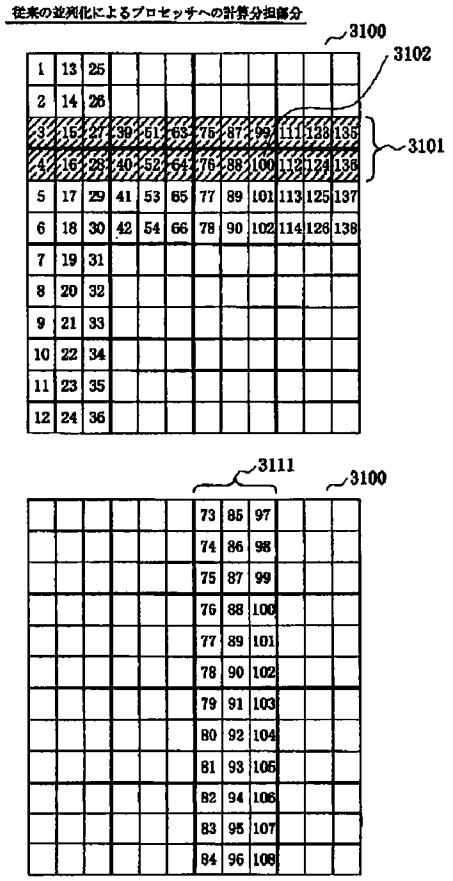
並列化後のプログラム　2720

```
2900:real a(3,3,4,4),b,c
2901:call fork
2902:do k = 1,10
2903:   call barrier
2910:   do j2 = 1,1
2911:   do j1 = 2,3
2912:   do i1 = 1,3
2913:     a(i1,j1,mype,j2)=a(i1,j1,mype,j2)+ &
2914:         a(i1,j1-1,mype,j2)*b
2915:   enddo;enddo;enddo
2920:   do j2 = 2,4
2921:   do j1 = 1,1
2922:   do i1 = 1,3
2923:     a(i1,j1,mype,j2)=a(i1,j1,mype,j2)+ &
2924:         a(i1,j1+2,mype,j2-1)*b
2925:   enddo;enddo;enddo
2930:   do j2 = 2,4
2931:   do j1 = 2,3
2932:   do i1 = 1,3
2933:     a(i1,j1,mype,j2)=a(i1,j1,mype,j2)+ &
2934:         a(i1,j1-1,mype,j2)*b
2935:   enddo;enddo;enddo
2940:   call barrier
2941:   do j1 = 1,3
2950:   do i2 = 1,1
2951:   do i1 = 2,3
2952:     a(i1,j1,i2,mype)=a(i1,j1,i2,mype)+ &
2953:         a(i1-1,j1,i2,mype)*c
2954:   enddo;enddo
2960:   do i2 = 2,4
2961:   do i1 = 1,1
2962:     a(i1,j1,i2,mype)=a(i1,j1,i2,mype)+ &
2963:         a(i1+2,j1,i2-1,mype)*c
2964:   enddo;enddo
2970:   do i2 = 2,4
2971:   do i1 = 2,3
2972:     a(i1,j1,i2,mype)=a(i1,j1,i2,mype)+ &
2973:         a(i1-1,j1,i2,mype)*c
2974:   enddo;enddo
2980:   enddo
2981:   call update(b,c)
2982:enddo
2983:call join
```

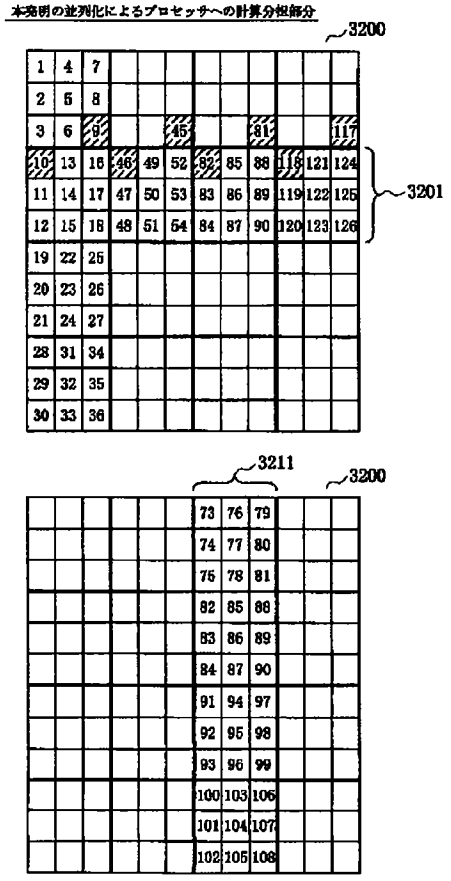
【図２７】


第２実施例に対する構成
ソース
プログラム
2710
2700
並列化コンパイラ
構文解析部
101
ループ並列化解析部
2701
データ分散形状決定部
2702
形状収集部
103
形状種別数判定部
104
配列細分化部
105
配列高次元化部
106
107
プログラム変換部
ループ変換部
1071
添字変更部
1072
プログラム並列化部
2703
コード生成部
109
120
辞書
130
中間語
140
データ
分散情報
150
ループ
テーブル
並列化
プログラム
2720
制御の流れ
データの流れ

【図３１】



【図３２】



フロントページの続き

(72)発明者　廣岡　孝志
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

Ｆターム(参考)　5B045 GG11
5B081 CC28 CC32